# ВВЕДЕНИЕ

Информацией владеют и используют её все люди без исключения. Каждый человек решает для себя, какую информацию ему необходимо получить, какая информация не должна быть доступна другим и т.д. Человеку легко, хранить информацию, которая у него в голове, а как быть, если информация занесена в «мозг машины», к которой имеют доступ многие люди.

Для предотвращения потери информации разрабатываются различные механизмы её защиты, которые используются на всех этапах работы с ней. Защищать от повреждений и внешних воздействий надо и устройства, на которых хранится секретная и важная информация, и каналы связи.

Повреждения могут быть вызваны поломкой оборудования или канала связи, подделкой или разглашением секретной информации. Внешние воздействия возникают как в результате стихийных бедствий, так и в результате сбоев оборудования или кражи.

Для сохранения информации используют различные способы защиты:

* безопасность зданий, где хранится секретная информация;
* контроль доступа к секретной информации;
* разграничение доступа;
* дублирование каналов связи и подключение резервных устройств;
* криптографические преобразования информации;

Создание и совершенствование компьютеров привело и продолжает приводить к созданию новых технологий в различных сферах научной и практической деятельности. Одной из таких сфер стало образова­ние – процесс передачи систематизированных знаний, навыков и умений от одного поколения к другому. Будучи само по себе мощной информационной сферой, и владея опытом использования различных классических (не компьютерных) инфор­мационных систем, образование быстро откликнулось на возможности современ­ной техники. Обучающая контролирующая программа – это системы, помогающие осваивать новый материал, производящие контроль знаний, помогающие преподавателям готовить учебный материал.

Одной из таких программ является обучающая контролирующая программа «Защита компьютеной информации».

Задачей данного курсового проекта является написание обучающей контролирующей программы, позволяющей просматривать темы о защите компьютеной информации, а также произвести контроль знаний по изучиной теме. В данной программе реализованы следующие функции:

* просмотр информации;
* прохождения тестов по изучиным темам;
* просмотр статистики пройденых тем.

Пояснительная записка состоит из четырёх разделов, содержащих необходимую информацию по организации и эксплуатации программного продукта.

Первый раздел «Анализ задачи» содержит описание предметной области курсового проекта, а также перечисляются требования к программе, описывается схема «Сущность-связь» и инструменты разработки.

Второй раздел «Проектирование задачи» описывает реляционную модель данных, структуру базы данных, входную и выходную информацию, а также структуру меню.

Третий раздел «Реализация» содержит описание организации функций добавления, редактирования, удаления и поиска, а также описание всех элементов управления, полное и функциональное тестирование.

Четвёртый раздел «Применение» содержит описание назначения программы. Также описывается разграничение доступа к информации, средства защиты и структура справочной системы.

Заключение содержит итоги проделанной работы.

**1 АНАЛИЗ ЗАДАЧИ**

**1.1 Описание предметной области**

Появление новых информационных технологий и развитие мощных компьютерных систем хранения и обработки информации повысили уровни защиты информации и вызвали необходимость в том, чтобы эффективность защиты информации росла вместе со сложностью архитектуры хранения данных. Так постепенно защита экономической информации становится обязательной: разрабатываются всевозможные документы по защите информации; формируются рекомендации по защите информации; даже проводится ФЗ о защите информации, который рассматривает проблемы защиты информации и задачи защиты информации, а также решает некоторые уникальные вопросы защиты информации.

Таким образом, угроза защиты информации сделала средства обеспечения информационной безопасности одной из обязательных характеристик информационной системы.

На сегодняшний день существует широкий круг систем хранения и обработки информации, где в процессе их проектирования фактор информационной безопасности Российской Федерации хранения конфиденциальной информации имеет особое значение. К таким информационным системам можно отнести, например, банковские или юридические системы безопасного документооборота и другие информационные системы, для которых обеспечение защиты информации является жизненно важным для защиты информации в информационных системах.

В связи с тем, что в мире существует большое количество информации о безапасности, в которых содержится большое количество видов и методов безапасности, было решено написать программу, позволяющую просматривать всю информацию о безапасности без особых усилий. В данной программе могут быть автоматизированы следующие функции:

* добавление информации о защите компьютерной информации;
* удаление информации матерьала;
* поиск информации по заданным критериям;
* прохождения тестов на каждую изучиную тему;
* просмотр информации, которыя содержатся в удобной для восприятия форме.

Обучающая контралирующая программа «Защита компьютерной информации» предназначена для тех людей, которые хотят больше узнать о безапасности информации. Также данную программу могут использовать для поиска интересующих их методов или видов защиты компьютерной безапасности.

* 1. **Модель данных**

Следующий этап в создании информационной системы – проектирование информационных структур. Результатом этого должна стать логическая модель данных, представляющая собой графическое изображение структуры информационной системы: названия таблиц с именами входящих в них полей и типами связей между полями таблиц с данными.

Реляционная модель данных – это логическая модель данных, прикладная теория построения баз данных, которая является приложением к задачам обработки данных таких разделов математики как теории множеств и логика первого порядка.

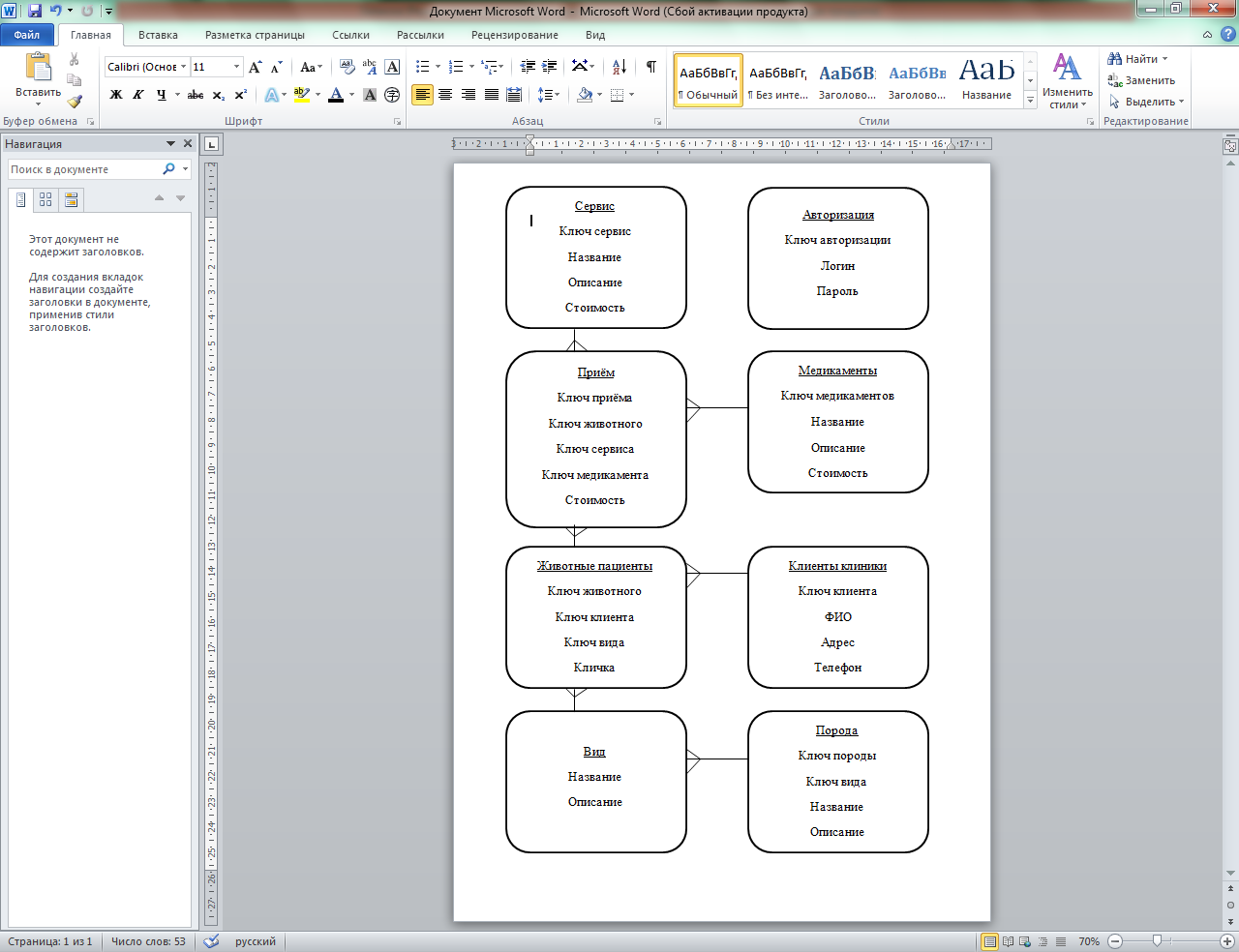


Рисунок 1 – Логическая модель данных

Рассмотрим каждый из структурных элементов и выделим сущности и их атрибуты.

Таблица 1 Сущности и их атрибуты

|  |  |  |
| --- | --- | --- |
| Сущность | Атрибут | Тип данных |
| Сервис | Ключ сервиса | Числовой, ненулевое значение |
| Название | Текст, ненулевое значение |
| Описание | Текст, ненулевое значение |
| Стоимость | Денежный |
| Приём | Ключ приёма | Числовой, ненулевое значение |
| Ключ животного | Числовой, ненулевое значение |
| Ключ сервиса | Числовой, ненулевое значение |
| Ключ препарата | Числовой, ненулевое значение |
| Стоимость | Денежный |
| Животные-Пациенты | Ключ животного | Числовой, ненулевое значение |
| Ключ клиента | Числовой, ненулевое значение |
| Ключ вида | Числовой, ненулевое значение |
| Кличка | Текст, ненулевое значение |
| Вакцинация | Текст, ненулевое значение |
| Вид | Ключ вида | Числовой, ненулевое значение |
| Название | Числовой, ненулевое значение |
| Сущность | Атрибут | Тип данных |
| Вид | Описание | Числовой, ненулевое значение |
| Авторизация | Ключ авторизации | Числовой, ненулевое значение |
| Логин | Числовой, ненулевое значение |
| Пароль | Числовой, ненулевое значение |
| Препараты | Ключ препарата | Числовой, ненулевое значение |
| Название | Текст, ненулевое значение |
| Описание | Текст, ненулевое значение |
| Стоимость | Денежный |
| Порода | Ключ породы | Числовой, ненулевое значение |
| Ключ вида | Числовой, ненулевое значение |
| Название | Текст, ненулевое значение |
| Описание | Текст, ненулевое значение |

* 1. **Инструменты разработки**

Для создания данного курсового проекта была выбрана операционная система Windows 10 и объектно-ориентированная среда Visual Studia 2013.

Windows 10 — операционная система для персональных компьютеров, разработанная корпорацией Microsoft в рамках семейства Windows NT. После Windows 8 система получила номер 10, минуя 9.

Система призвана стать единой для разных устройств, таких как персональные компьютеры, планшеты, смартфоны, консоли Xbox One и пр. Доступна единая платформа разработки и единый магазин универсальных приложений, совместимых со всеми поддерживаемыми устройствами. Windows 10 поставляется в качестве услуги с выпуском обновлений на протяжении всего цикла поддержки. В течение первого года после выхода системы пользователи могут бесплатно обновиться до Windows 10 на устройствах под управлением легальных Windows 7, Windows 8.1 и Windows Phone 8.1. Среди значимых нововведений — голосовая помощница Кортана, возможность создания и переключения нескольких рабочих столов и др. Windows 10 — последняя «коробочная» версия Windows, все последующие версии будут распространяться исключительно в цифровом виде.

В качестве системы программирования была выбрана интегрированная система Visual Studio 2013. Microsoft Visual Studio — линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight.

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и как отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (как, например, Subversion и Visual SourceSafe), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования) или инструментов для прочих аспектов процесса разработки программного обеспечения (например, клиент Team Explorer для работы с Team Foundation Server).

В состав технических средств должен входить персональный компьютер, удовлетворяющий следующим минимальным требованием:

1. процессор AMD A8-6600K;
2. объем ОЗУ 8 Гб;
3. объем места на HDD 2000 Гб;
4. видеоподсистема 1280 x 1024 точек с глубиной цвета 64 bit;
5. «ОС Windows 7».

**1.4 Требования к приложению**

В программе должны быть реализованы две группы пользователей: «Администратор» и «Пользователь». Пользователи могут свободно пользоваться программой.

Для пользователя «Пользователь» должны быть реализованы следующие функции:

* просмотр информации о зоопарках;
* просмотр фотографий и видеоматериалов;
* поиск интересующей информации.

Для пользователя «Администратор» должны быть реализованы следующие функции:

* добавление информации о зоопарках, животных, видах животных и материалов к ним;
* редактирование информации о зоопарках, животных и видах животных;
* удаление информации о зоопарках, животных, видах животных и материалов к ним.

В курсовом проекте должна быть предусмотрена справочная система, позволяющая без проблем ориентироваться в программе и полностью освоить её функционал. Интерфейс программы должен быть приемлемым и понятным пользователю.

Должны выдаваться сообщения о неправильно введенном пароле, не введенных данных при добавлении любой информации, сообщение о введении неверных данных. Также должны проводиться проверки на вводимые данные, к примеру, такие как ограничение на ввод определённых символов при добавлении, редактировании информации. При добавлении информации должна быть проверка на повторяющиеся записи.

**2 ПРОЕКТИРОВАНИЕ ЗАДАЧИ**

**2.1 Организация данных**

Для создания базы данных информационно-справочной системы использовалась встроенная в среду разработки программа Database Desktop, позволяющая быстро создавать базы данных. В программе Database Desktop был выбран тип базы данных Paradox 7, которая позволяет манипулировать данными различных форматов. Достоинствами Database Desktop являются: легкость создания таблиц базы данных, удобная обработка данных содержащихся в таблицах, понятный интерфейс утилиты.

База данных состоит из девяти таблиц, структура которых показана в таблицах 1 – 9.

Таблица 1 – Структура таблицы «Materials»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Ключевое поле | Описание поля |
| Id\_Material | Autoincrementation | + | Код материала |
| Id\_Link | Integer |  | Код связи |
| Path | Alpha (100) |  | Путь к материалу |
| Type | Integer |  | Тип материала |

Таблица 2 – Структура таблицы «Zoos»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Ключевое поле | Описание поля |
| Id\_Zoo | Autoincrementation | + | Код зоопарка |
| Id\_Country | Integer |  | Код страны |
| Title | Alpha (50) |  | Название зоопарка |
| City | Alpha (50) |  | Город |
| Area | Integer |  | Площадь |
| ZooDesciption | Memo |  | Описание зоопарка |

Таблица 3 – Структура таблицы «Species»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Ключевое поле | Описание поля |
| Id\_Species | Autoincrementation | + | Код вида |
| Id\_Climate | Integer |  | Код климата |
| Id\_Habitat | Integer |  | Код среды обитания |
| Id\_FoodType | Integer |  | Код типа питания |
| Name | Alpha (100) |  | Название вида |
| Description | Memo |  | Описание вида |

Таблица 4 – Структура таблицы «Pictures»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Ключевое поле | Описание поля |
| Id\_Picture | Autoincrementation | + | Код картинки |
| Id\_Zoo | Integer |  | Код зоопарка |
| Path | Alpha (100) |  | Путь к фотографии |

Таблица 5 – Структура таблицы «Countries»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Ключевое поле | Описание поля |
| Id\_country | Autoincrementation | + | Код страны |
| Country | Integer |  | Название страны |

Таблица 6 – Структура таблицы «Link»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Ключевое поле | Описание поля |
| Id\_Link | Autoincrementation | + | Код связи |
| Id\_Zoo | Integer |  | Код зоопарка |
| Id\_Species | Integer |  | Код вида |
| RegDate | Date |  | Дата регистрации |
| Note | Memo |  | Примечание |
| Name | Alpha (30) |  | Имя животного |

Таблица 7 – Структура таблицы «Habitats»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Ключевое поле | Описание поля |
| Id\_Habitat | Autoincrementation | + | Код среды обитания |
| Habitat | Alpha (100) |  | Среда обитания |

Таблица 8 – Структура таблицы «FoodTypes»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Ключевое поле | Описание поля |
| Id\_FoodType | Autoincrementation | + | Код типа питания |
| FoodType | Alpha (30) |  | Тип питания |

Таблица 9 – Структура таблицы «Climates»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Ключевое поле | Описание поля |
| Id\_Climate | Autoincrementation | + | Код климата |
| Climate | Alpha (30) |  | Климат |

Входная информация – данные, поступающие на вход задачи и используемые для её решения. К входной информации в данной программе относится: пользовательский ввод, добавленная информация, критерии поиска.

Выходная информация – может быть представлена в виде документов, кадров на экране монитора, информации в базе данных, выходного сигнала устройству управления. К выходной информации в данной программе относится: распечатанная информация, результаты поиска.

**2.2 Концептуальный прототип**

В программе будет реализовано разделение пользователей. Меню для каждого пользователя будет формироваться по отдельности.

Обычный пользователь будет иметь следующие кнопки: «Просмотр информации», «Помощь» и «Выход».

При нажатии на кнопку «Просмотр информации» отображается форма, на которой отображена вся информация по зоопаркам, также на этой форме расположены три кнопки: «Найти», «Экспорт в Word» и «Назад». При нажатии на кнопку «Найти» откроется форма с полями для поиска и кнопками:

* «Найти» –при нажатии на неё произойдёт поиск по заданным критериям;
* «Сбросить» – при нажатии на неё сбрасываются все критерии поиска, а также результат поиска;
* «Назад»– при нажатии на неё скрывается форма поиска.

При нажатии на кнопку «Экспорт в Word» произойдёт экспорт в «Microsoft office Word», после чего откроется окно редактора «MS Word» со всей информацией о выбранном зоопарке. При нажатии на кнопку «Назад» скроется форма просмотра информации и откроется форма, отображающая меню пользователя.

Пользователь «Администратор» будет иметь следующие кнопки: «Добавление», «Редактирование», «Удаление», «Обратно» и«Выход». При нажатии на кнопку «Добавление» откроется форма с одиннадцатью кнопками: девять кнопок «Добавить» и две «Загрузить». Каждая кнопка «Добавить» соответствует своей ячейке и выполняет соответствующее действие. Кнопки «Загрузить» выполняют загрузку фотографий и видеоматериалов. При нажатии на кнопку «Редактирование» откроется форма со следующими кнопками:

* «Редактировать»– при нажатии на неё откроется форма для редактирования выбранной информации;
* «Обратно»– при нажатии возвращает на форму выбора действия.

При нажатии на кнопку «Удаление» откроется форма со следующими кнопками:

* «Удалить» – при нажатии на неё откроется форма для удаления выбранной информации;
* «Обратно» – при нажатии возвращает на форму выбора действия.

При нажатии на кнопку «Обратно» происходит выход из режима «Администратор» и возврат на меню пользователя. При нажатии на кнопку «Выход» программа завершает свою работу.

**2.3 Система справочной информации**

Справочная система будет создана в программе HTML Help Workshop.

HTML Help Workshop – удобная программа для создания файлов справки, книг или журналов в форматах HTML-справка (CHM). Страницы справок создаются в удобном редакторе, большим плюсом которого является то, что он не использует сторонних компиляторов, при этом имеет сравнительно небольшой размер. Особенности программы:

* встроенный визуальный HTML-редактор;
* редактор оглавлений и значков;
* визуальный дизайнер CHM [3].

Данная программа не сложна в использовании. Она позволяет создать справочную систему в несколько этапов, что не занимает большого времени. Окно справки имеет меню и панель инструментов, позволяющее работать со справочными файлами, а также содержит указатель на тему и поиск.

Справочная система для информационно-справочной системы «Зоопарки Мира» состоит из следующих разделов:

* «Добро пожаловать!» – данный раздел служит страницей приветствия пользователя;
* «Как войти в систему?»– описывает процесс входа в систему;
* «Как просмотреть информацию?»– описывает процесс просмотра информации;
* «Как добавлять?»– описывает процесс добавления какой-либо информации;
* «Как удалять?»– описывает процесс удаления какой-либо информации;
* «Как редактировать?»– описывает процесс редактирования какой-либо информации.

**3 РЕАЛИЗАЦИЯ**

**3.1 Функции: логическая и физическая организация**

В данном курсовом проекте реализованы следующие функции: добавление, удаление, редактирование и поиск.

Добавление осуществляется следующим образом: в первую очередь с помощью запроса «Select» выбирается нужная информация и проверяется на повторяющиеся записи. Затем с помощью запроса «Insert into» запись заносится в таблицу. Затем выводится сообщение об успешном добавлении с помощью специального диалога, после чего обновляется форма и очищаются поля.

Редактирование осуществляется следующим образом: вначале выбирается необходимая запись, и вводятся новые данные. Затем новые данные проверяются на наличие повторяющихся данных в таблице. После чего с помощью запроса «Update» запись обновляется, выводится сообщение с помощью специального диалога, также обновляется форма и очищаются поля.

Удаление осуществляется следующим образом: в первую очередь выбирается необходимая запись, после чего появляется диалоговое окно с запросом на подтверждение удаления. После подтверждения с помощью запроса «Delete»происходит удаление записи, вывод сообщения с помощью специального диалога и обновляется форма.

Поиск осуществляется следующим образом: в первую очередь вводятся критерии поиска, после чего с помощью запроса делается выборка в соответствии с условием, зависящим от критериев поиска. Затем данные отображаются на форме.

Более подробный программный код данных функций показан в приложении А.

3.2 Функции и элементы управления

Проект – это структура управления, которая позволяет взаимодействовать программному коду и визуальным объектам.

Модуль – это объект, который действует как связующий компонент приложения, фиксирующий взаимосвязи в программе. Данный проект содержит в себе модули:

* «Add.pas»– содержит форму добавления;
* «AdminMenu.pas»–содержит форму для выбора действий предназначенных для администратора;
* «Delete.pas»–содержит форму удаления;
* «FirstDelete.pas»–содержит форму удаления примитивных данных;
* «FirstUpdate.pas»–содержит форму обновления примитивных данных;
* «MainMenu.pas»–содержит форму выбора действий пользователя;
* «Message.pas»–содержит диалоговое окно, предназначенное для вывода сообщений;
* «Password.pas»–содержит форму для ввода пароля;
* «Question.pas»–содержит диалоговое окно для вывода сообщений в виде вопросов;
* «ScreenSaver.pas»–содержит форму с заставкой;
* «Search.pas»–содержит форму для поиска;
* «SecondDelete.pas» – содержит форму для удаления сложных видов данных;
* «SecondUpdate.pas» – содержит форму для редактирования сложных видов данных;
* «ShowMaterials.pas» – содержит форму для просмотра материалов;
* «ShowZoos.pas» – содержит форму для просмотра информации о зоопарка;
* «MenuUnit.pas» – содержит класс для работы с меню;
* «TimerUnit.pas» – содержит функции для плавного перехода между формами;
* «Update.pas»–содержит форму для редактирования данных.

Полный проект программной системы представляет собой совокупность моделей логического и физического уровней, которые должны быть согласованы между собой. При разработке были использованы следующие компоненты:

TPaintBox – это полотно для рисования. Используется для задания кнопок на форме. Все кнопки рисуются программно и их поведения также задаётся программно. Для данного элемента управления разработаны следующие функции: «Реакция на нажатие мышью», «Реакция на движение мышью», «Реакция на запрос рисования». На всех формах эти функции именуются одинаково, то есть следующим образом:

* Procedure MenuBoxPaint (Sender:TObject) – поведение объекта при прорисовке;
* procedure MenuBoxMouseMove (Sender: TObject; Shift: TShiftState; X,Y: Integer)– задаёт поведение кнопкам;
* procedure MenuBoxClick (Sender: TObject) – задаёт поведение кнопок при нажатии.

TDBGrid– компонент позволяющий отображать данные из таблиц в удобном виде. Для данного элемента управления разработаны следующие функции:

* procedure TSecondUpdateFm.DBGrdZooDrawColumnCell (Sender: TObject; constRect: TRect; DataCol: Integer; Column: TColumn; State: TGridDrawState) – используется для перерисовки таблицы зоопарки;
* procedure TSecondUpdateFm.DBGrdZooColExit (Sender: TObject), procedure TSecondUpdateFm.DBGrdAnimColExit (Sender: TObject), procedure TSecondUpdateFm.DBGrdSpeciesColExit (Sender: TObject) – используется для завершения редактирования записи;
* procedure TSecondUpdateFm.DBGrdAnimDrawColumnCell (Sender: TObject; constRect: TRect; DataCol: Integer; Column: TColumn; State: TGridDrawState) – используется для перерисовки таблицы животные;
* procedure TSecondUpdateFm.DBGrdZooCellClick (Column: TColumn)– используется для вызова поля предназначенного для редактирования зоопарков;
* procedure TSecondUpdateFm.DBGrdAnimCellClick (Column: TColumn) – используется для вызова поля предназначенного для редактирования животных;
* procedure TSecondUpdateFm.DBGrdSpeciesCellClick (Column: TColumn) – используется для вызова поля предназначенного для редактирования видов;
* procedure TShowZoosFm.dbGrdZooCellClick (Column: TColumn) – используется для выбора животных соответствующих выбранному зоопарку;
* procedure TShowZoosFm.dbGrdAnimCellClick (Column: TColumn) – используется для выбора картинок относящихся к выбранному животному.

TLabeledEdit– компонент совмещающий в себе качества обычного labelи edit. Для данного элемента управления разработаны следующие функции:

* procedure TAddFm.LetKeyPress (Sender: TObject; var Key: Char), procedure TSecondUpdateFm.LetKeyPress (Sender: TObject; var Key: Char) –запрещает ввод цифр;
* procedure TAddFm.NumKeyPress (Sender: TObject; var Key: Char), procedure TSecondUpdateFm.NumKeyPress (Sender: TObject; var Key: Char) – запрещает ввод букв;

TDBLookupComboBox – компонент напоминает обычный «ComboBox», однако данные берёт с таблицы. Для данного элемента управления разработаны следующие функции:

* procedure TAddFm.dbCbCountry2Click (Sender: TObject), procedure TAddFm.dbCbCountry3Click (Sender: TObject), procedure TAddFm.dbCbCountry4Click (Sender: TObject), procedure TAddFm.dbCbZoo3Click (Sender: TObject), procedure TShowZoosFm.dbCbCountryClick (Sender: TObject), procedure TSecondDeleteFm.dbCbAnim2Click (Sender: TObject), procedure TSecondDeleteFm.dbCbZoo2Click (Sender: TObject), procedure TSecondDeleteFm.dbCbZoo3Click (Sender: TObject), procedure TSecondDeleteFm.dbCbZoo4Click (Sender: TObject) – осуществляют фильтрацию данных по выбранной записи;

TTimer– компонент, позволяющий после истечения заданного интервала выполнять заданную функцию. Все таймеры в данном проекте выполняют одну и туже функцию. Функция заключается в плавном переходе между формами.

**3.3 Функциональное тестирование**

После запуска программы на экране появилась заставка, которая изображена на рисунке Б.1. Через некоторое время заставка закрылась и появилась форма выбора действия, которая показана на рисунке Б.2.

Затем нажали на кнопку «Просмотр информации» и появилась форма, изображённая на рисунке Б.3. На данной форме просмотрели информацию, выбрав в раскрывающемся списке страну, о зоопарках, животных, видах животных, также их фотографии и видеоматериалы. При просмотре можно произвести поиск и экспортировать в редактор Microsoft Office Word. После просмотра нажали на кнопку «Обратно» и вернулись в меню, показанное на рисунке Б.2.

На форме выбора действия нажали на кнопку «Помощь» и открылось окно со справочной системой, которое показано на рисунке Б.4.

Для входа в систему под пользователем «Администратор» нажали сочетание клавиш «Ctrl + F4» и появилась форма, для ввода пароля показанная на рисунке Б.5. На форме ввели пароль и нажали на кнопку «Готово». При вводе неправильного пароля появилось сообщение о том, что введен неверный пароль, изображенное на рисунке Б.6. При вводе правильного пароля открылась форма администратора, изображённая на рисунке Б.7.

Для добавления данных нажали на кнопку «Добавление», после чего появилась форма для добавления данных, показанная на рисунке Б.8.

Для добавления страны в поле «Введите страну: » ввели название страны и нажали на кнопку «Добавить». Если поле оставили пустым или ввели уже существующую информацию, появилось сообщение Б.9 и Б.10 соответственно. При правильном вводе названия она добавилась в базу данных.

Для добавления среды обитания в поле «Введите среду обитания: » ввели название среды обитания и нажали на кнопку «Добавить». Если поле оставили пустым или ввели уже существующую информацию, появилось сообщение Б.9 и Б.10 соответственно. При правильном вводе названия она добавилась в базу данных.

Для добавления типа питания в поле «Введите тип питания: » ввели название типа питания и нажали на кнопку «Добавить». Если поле оставили пустым или ввели уже существующую информацию, появилось сообщение Б.9 и Б.10 соответственно. При правильном вводе названия она добавилась в базу данных.

Для добавления климата в поле «Введите климат: » ввели название климата и нажали на кнопку «Добавить». Если поле оставили пустым или ввели уже существующую информацию, появилось сообщение Б.9 и Б.10 соответственно. При правильном вводе названия она добавилась в базу данных.

Для добавления зоопарка в выпадающем списке «Выберите страну: » выбрали уже добавленную страну, заполнили поля «Введите название: », «Введите площадь: », «Введите город: », «Введите информацию: » и нажали на кнопку «Добавить». Если поля оставили пустыми или ввели уже существующую информацию, появилось сообщение Б.9 и Б.10 соответственно. При правильном вводе данных они добавились в базу данных.

Для добавления вида животного заполнили поле «Введите название: », в соответствующих раскрывающихся списках выбрали среду обитания, климат, тип питания, ввели описание и нажали на кнопку «Добавить». Если поля оставили пустыми или ввели уже существующую информацию, появилось сообщение Б.9 и Б.10 соответственно. При правильном вводе данных они добавились в базу данных.

Для добавления животного в зоопарк в соответствующих раскрывающихся списках выбрали страну, зоопарк, вид животного, дату регистрации, в поле «Введите имя животного: » ввели имя животного, в поле «Введите примечание: » – его описание и нажали на кнопку «Добавить». Если поля оставили пустыми или ввели уже существующую информацию, появилось сообщение Б.9 и Б.10 соответственно. При правильном вводе данных они добавились в базу данных.

Для добавления фотографий к зоопарку в раскрывающихся списках «Выберите страну: », «Выберите зоопарк: » соответственно выбрали страну и название зоопарка. Нажали на кнопку «Загрузить» и открылось окно выбора фотографии, показанное на рисунке Б.11. Выбрали фотографию и нажали на кнопку «Открыть». На форме отобразилась, выбранная фотография. Далее нажали на кнопку «Добавить». Если поля оставили пустыми или ввели уже существующую информацию, появилось сообщение Б.9 и Б.10 соответственно. При правильном вводе данных они добавились в базу данных.

Для добавления фотографий и видеоматериалов к животному выбрали в раскрывающихся списках соответственно страну, зоопарк, животное, тип материала и нажали на кнопку «Загрузить». Открылось окно выбора фотографии или видео, показанное на рисунке Б.11. Выбрали фотографию и нажали на кнопку «Открыть». На форме отобразилась, выбранная информация. Далее нажали на кнопку «Добавить». Если поля оставили пустыми или ввели уже существующую информацию, появилось сообщение Б.9 и Б.10 соответственно. При правильном вводе данных они добавились в базу данных. Для возврата к меню администратора нажали на клавишу «Esc».

Для редактирования информации нажали на кнопку «Редактирование» и открылась форма, показанная на рисунке Б.12. Для редактирования страны выбрали пункт «Страны» и нажали кнопку «Редактировать». Затем открылась форма, изображённая на рисунке Б.13. В раскрывающемся списке выбрали страну, в поле «Введите новое название: » ввели новое название и нажали кнопку «Редактировать». После чего запись изменилась в базе данных. Для возврата на форму редактирования нажали кнопку «Обратно». Для редактирования типа питания выбрали пункт «Типы питания» и нажали на кнопку «Редактировать». Затем открылась форма, изображённая на рисунке Б.14. В списке выбрали тип питания, в поле «Введите новое название: » ввели новое название и нажали на кнопку «Редактировать». После чего запись изменилась в базе данных. Для возврата на форму редактирования нажали кнопку «Обратно». Для редактирования климата выбрали пункт «Климаты» и нажали на кнопку «Редактировать». Затем открылась форма, показанная на рисунке Б.15. В списке выбрали климат, в поле «Введите новое название: » ввели новое название и нажали на кнопку «Редактировать». После чего запись изменилась в базе данных. Для возврата на форму редактирования нажали кнопку «Обратно». Для редактирования среды обитания выбрали пункт «Среды обитания» и нажали на кнопку «Редактировать». Затем открылась форма, показанная на рисунке Б.16. В списке выбрали среду обитания, в поле «Введите новое название: » ввели новое название и нажали на кнопку «Редактировать», После чего запись изменилась в базе данных. Для возврата на форму редактирования нажали кнопку «Обратно». Для редактирования данных о зоопарке, животных или видах животных выбрали пункт «Иное» и нажали на кнопку «Редактировать». Затем открылась форма, изображённая на рисунке Б.17. Чтобы на ней редактировать нажимаем на любое поле в группах «Редактирование зоопарков», «Редактирование животных», «Редактирование видов», после чего вместо поля отображается раскрывающийся список или поле ввода с возможностью редактирования. После внесения изменений, для их подтверждения нажимаем клавишу «Enter» или нажимаем на любое другое поле. Для возврата на форму редактирования нажали кнопку «Обратно». Для возврата к меню администратора нажали на кнопку «Обратно».

Для удаления данных нажали на кнопку «Удаление» и открылась форма удаления, изображённая на рисунке Б.18. Для удаления страны выбрали пункт «Страны» и нажали на кнопку «Удалить». Затем открылась форма, показанная на рисунке Б.19. В списке выбрали страну и нажали на кнопку «Удалить». После чего появилось окно подтверждения удаления, оно изображено на рисунке Б.20. Нажали на кнопку «Да». Затем запись удалилась из базы данных. Для возврата на форму удаления данных нажали кнопку «Обратно». Для удаления типа питания выбрали пункт «Типы питания» и нажали на кнопку «Удалить». Затем открылась форма, показанная на рисунке Б.21. В списке выбрали тип питания и нажали на кнопку «Удалить». После чего появилось окно подтверждения удаления, оно изображено на рисунке Б.20. Нажали на кнопку «Да», затем запись удалилась из базы данных. Для возврата на форму удаления данных нажали кнопку «Обратно». Для удаления климата выбрали пункт «Климаты» и нажали на кнопку «Удалить». Затем открылась форма, показанная на рисунке Б.22. В списке выбрали климат и нажали на кнопку «Удалить». После чего появилось окно подтверждения удаления, оно изображено на рисунке Б.20. Нажали на кнопку «Да», затем запись удалилась из базы данных. Для возврата на форму удаления данных нажали кнопку «Обратно». Для удаления среды обитания выбрали пункт «Среды обитания» и нажали на кнопку «Удалить». Затем открылась форма, изображённая на рисунке Б.23. В списке выбрали среду обитания и нажали на кнопку «Удалить». После чего появилось окно подтверждения удаления, оно изображено на рисунке Б.20. Нажали на кнопку «Да», затем запись удалилась из базы данных. Для возврата на форму удаления данных нажали кнопку «Обратно». Для удаления зоопарков, животных, видов животных, фотографий и видео выбрали пункт «Иное» и нажали на кнопку «Удалить». После чего открылась форма, показанная на рисунке Б.24. Для удаления зоопарка выбрали зоопарк и нажали на кнопку «Удалить». После подтверждения удаления запись удалилась из базы данных. Для удаления животного выбрали зоопарк, животное и нажали на кнопку «Удалить». После подтверждения удаления запись удалилась из базы данных. Для удаления фотографии выбрали зоопарк, фотографию и нажали на кнопку «Удалить». После подтверждения удаления запись удалилась из базы данных. Для удаления фотографий и видео выбрали зоопарк, животное, фотографию или видео и нажали на кнопку «Удалить». После подтверждения удаления запись удалилась из базы данных. Для возврата на форму удаления нажали кнопку «Обратно». Для возврата к меню администратора нажали на кнопку «Обратно».

Для завершения работы программы нажали на кнопку «Выход».

**3.4 Полное т**е**стирование**

Полное тестирование провели на примере добавления зоопарка с неверно заполненными данными и их последующим редактированием. Для этого зашли в программу под пользователем «Администратор», затем в меню администратора нажали на кнопку «Добавление». После чего появилась форма добавления, показанная на рисунке Б.8, где в блоке «Добавление зоопарка:» выполнили следующие действия:

* выбрали страну: «Беларусь»;
* ввели название: «Гродненский зоопарк»;
* ввели площадь: «10»;
* ввели город: «Гродно»;
* ввели информацию: «Очень красивый зоопарк».

После всех действий нажали на кнопку «Добавить». После чего появилось сообщение, показанное на рисунке Б.25.

Для того чтобы отредактировать добавленную информацию о зоопарке необходимо: нажать клавишу «Esc», затем в меню администратора нажать на кнопку «Редактирование», после чего появиться форма, изображённая на рисунке Б.12. На этой форме выбрали пункт «Иное» и нажали на кнопку «Редактировать». Открылась форма, показанная на рисунке Б.17. На форме выполнили следующие действия:

* в таблице «Редактирование зоопарков: » нажали на поле «Площадь»;
* в появившееся поле ввели значение «1000»;
* нажали «Enter».

После нажатия на клавиатуре клавиши «Enter» запись изменилась в базе данных. Вернувшись в меню пользователя, нажали на кнопку «Просмотр информации». Отрылась форма для просмотра информации, она показана на рисунке Б.28.

**4 ПРИМЕНЕНИЕ**

**4.1 Назначение**

Информационно-справочная система «Зоопарки мира» предназначена в первую очередь для детей, желающих узнать информацию:

* о зоопарках всего мира;
* о животных, обитающих в зоопарках;
* об особенностях содержания тех или иных видов животных.

Также ей могут воспользоваться все желающие, которые хотят узнать что-либо о зоопарках и о мире животных.

В данной программе автоматизированы следующие функции:

* просмотр информации о зоопарках, животных, видах животных, а также же их фотографий и видеоматериалов;
* добавление, редактирование, удаление информации о зоопарках, животных, видах животных, их фотографий и видеоматериалов;
* поиск информации по заданным критериям.

Назначение данной программы заключается в том, чтобы быстро и в полном объеме предоставить нужную информацию пользователю.

**4.2 Инсталляция**

Для того чтобы установить данную программу себе на компьютер необходимо выделить все файлы программы и переместить их в удобное для использования место.

Для того чтобы запустить программу необходимо дважды щёлкнуть по файлу «Zoos.exe», после чего программа запустится.

Для правильной работы программы на компьютере должны быть установлены следующие программы: Windows 7, драйвера соответствующие вашей конфигурации компьютера.

Для того чтобы войти под пользователем «Администратор» необходимо в меню нажать сочетание клавиш «Ctrl + F4», после чего в открывшемся окне ввести пароль «hiadmin». Затем программа предоставит возможности администратора.

**4.3 Сообщения системы**

Как и в любой программе, в данной системе выдаются сообщения при произведении каких-либо действий.

Если при добавлении, редактировании, удалении не заполнить все поля, то появится сообщение, изображённое на рисунке Б.9.

На рисунке Б.6 показано сообщение, выдаваемое при вводе неверного пароля.

Если добавлять уже существующую информацию, то появится сообщение, показанное на рисунке Б.10.

При успешном добавлении записи появляется сообщение, которое показано на рисунке Б.25.

При успешном удалении информации появляется сообщение, показанное на рисунке Б.26.

**4.4 Обучение**

Любой пользователь может воспользоваться справочной системой для освоения данной программы. Для этого на форме меню пользователя необходимо выбрать пункт меню «Помощь». После нажатия на кнопку откроется окно справочной системы, показанное на рисунке Б.4. С помощью справки можно ознакомиться с основами пользования программой.

Для того чтобы распечатать информацию из справочной системы необходимо нажать на кнопку «Печать».

Также в справочной системе предусмотрен поиск. Для того чтобы им воспользоваться необходимо нажать на кнопку «Поиск». Для поиска необходимо ввести в первое поле ключевое слово или выбрать его из предоставленных в списке и нажать кнопку «Показать».

**ЗАКЛЮЧЕНИЕ**

Задачей данного курсового проекта являлась разработка информационно-справочной системы «Зоопарки мира». Программа выполнена в соответствии с условиями задачи. Для реализации данного проекта была выбрана операционная система Window 7. Она была выбрана из-за своей простоты и удобства использования, а также за обширные возможности. Программа была создана в среде разработки BorlandDelphi 7.0.

В ходе написания курсового проекта были разработаны следующие функции:

* просмотр информации о зоопарках и их жителей;
* просмотр видео про животных;
* добавление, удаление и редактирование информации о зоопарках, видах животных, животных, а также фотографий и видеоматериалов;
* поиск по заданным критериям.

Вся программа неоднократно тестировалась на правильность получаемых ответов, все возникающие ошибки были устранены в ходе работы. После проведения тестирования можно сказать, что программа удовлетворяет всем поставленным перед ней требованиям и является готовым программным средством.

Так как данная программа реализована с помощью программного средства BorlandDelphi 7.0, то в будущем программу можно усовершенствовать. Возможными усовершенствованиями могут быть:

* добавление сетевого режима;
* расширения информативности программы;
* более красочный интерфейс.

Смотря на программу можно с уверенностью сказать, что в данном курсовом проекте предусмотрены и реализованы все необходимые функции для удобного и успешного просмотра информации про «Зоопарки мира».

Данный проект может применяться в школах детьми для расширения кругозора, изучения видов животных обитающих зоопарках. Также программой могут пользоваться все желающие что-либо узнать про животных и зоопарки или просто просмотреть фотографии и видеоматериалы.

**ЛИТЕРАТУРА**

1. Левин, А.А. Работа с Windows / А.А. Левин. – Москва: Омега, 2005. – 215 с.
2. Фленов, М. Библия Delphi, 3-е издание / М. Фленов. – Петербург: БХВ-Петербург, 2011. – 1024с.
3. Осипов, Д. Профессиональное программирование / Д. Осипов. – Москва: Символ-плюс, 2006. – 630с.
4. Гласс, Р. Креативное программирование 2.0 / Р. Гласс. – Москва: Символ-плюс, 2009. – 352с.
5. Осипов, Д. Графика в проектах Delphi / Д. Осипов. – Москва: Символ-плюс, 2008. – 501с.
6. Чиртик, А. Delphi. Трюки и эффекты / А. Чиртик. – Санкт-Петербург: Питер, 2007 – 400с.
7. Понамарёв, В. Базы данных в Delphi 7. Самоучитель / В. Понамарёв. – Санкт-Петербург: Питер, 2003 – 224с.
8. Бобровский С. Delphi 7. Учебный курс / С. Бобровский. – Санкт-Петербург: Питер, 2004 – 736с.

**ПРИЛОЖЕНИЕ А**

**(обязательное)**

**Программный код**

unit Add;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, MenuUnit, ExtCtrls, MPlayer, ComCtrls, StdCtrls, DBCtrls, DB,

DBTables, ExtDlgs, jpeg;

type

TAddFm = class(TForm)

MenuBox: TPaintBox;

lbl1: TLabel;

bvl5: TBevel;

lbEdZooName: TLabeledEdit;

lbl5: TLabel;

dbCbCountry: TDBLookupComboBox;

lbEdArea: TLabeledEdit;

lbEdCity: TLabeledEdit;

memDescription: TMemo;

bvl6: TBevel;

lbl6: TLabel;

bvl7: TBevel;

lbl7: TLabel;

bvl8: TBevel;

lbl8: TLabel;

bvl9: TBevel;

lbl9: TLabel;

lbEdCountry: TLabeledEdit;

lbl2: TLabel;

bvl2: TBevel;

lbEdHabitat: TLabeledEdit;

lbl3: TLabel;

bvl3: TBevel;

lbEdFoodType: TLabeledEdit;

lbl4: TLabel;

bvl4: TBevel;

lbEdClimate: TLabeledEdit;

bvl1: TBevel;

lbEdSpeciesName: TLabeledEdit;

dbCbHabitat: TDBLookupComboBox;

dbCbFoodType: TDBLookupComboBox;

dbCbClimate: TDBLookupComboBox;

dbCbZoo: TDBLookupComboBox;

dbCbSpecies: TDBLookupComboBox;

memNote: TMemo;

dbCbZoo2: TDBLookupComboBox;

imgZoo: TImage;

lbEdAnimName: TLabeledEdit;

dbCbAnimal: TDBLookupComboBox;

cbType: TComboBox;

imgAnimal: TImage;

dtp: TDateTimePicker;

videoPanel: TPanel;

MP: TMediaPlayer;

lbl10: TLabel;

lbl11: TLabel;

lbl12: TLabel;

dbCbZoo3: TDBLookupComboBox;

lbl13: TLabel;

lbl14: TLabel;

lbl15: TLabel;

lbl16: TLabel;

lbl17: TLabel;

lbl18: TLabel;

lbl19: TLabel;

lbl20: TLabel;

lbl21: TLabel;

lbl22: TLabel;

imgBackGround: TImage;

qryAddZoo: TQuery;

qryAddAnimZoo: TQuery;

qryAddAnimMat: TQuery;

qryAddZooPic: TQuery;

qryAddSpecies: TQuery;

qryAddCountry: TQuery;

qryAddHabitat: TQuery;

qryAddFoodType: TQuery;

qryAddClimate: TQuery;

qryShowCountry: TQuery;

qryShowZoo: TQuery;

qryShowSpecies: TQuery;

qryShowZoo3: TQuery;

qryShowAnim: TQuery;

qryShowHabitat: TQuery;

qryShowClimate: TQuery;

qryShowFoodType: TQuery;

ds1: TDataSource;

ds2: TDataSource;

ds3: TDataSource;

ds4: TDataSource;

ds5: TDataSource;

ds6: TDataSource;

qryShowZoo2: TQuery;

ds7: TDataSource;

ds8: TDataSource;

ds9: TDataSource;

ShowHideTimer: TTimer;

dbCbCountry2: TDBLookupComboBox;

lbl23: TLabel;

qryShowCountry2: TQuery;

ds10: TDataSource;

dbCbCountry3: TDBLookupComboBox;

ds11: TDataSource;

qryShowCountry3: TQuery;

lbl24: TLabel;

dbCbCountry4: TDBLookupComboBox;

ds12: TDataSource;

qryShowCountry4: TQuery;

lbl25: TLabel;

dlgOpenVideo: TOpenDialog;

dlgOpenPic: TOpenPictureDialog;

memDescriptionSpec: TMemo;

procedure FormCreate(Sender: TObject);

procedure MenuBoxPaint(Sender: TObject);

procedure MenuBoxMouseMove(Sender: TObject; Shift: TShiftState; X,

Y: Integer);

procedure MenuBoxClick(Sender: TObject);

procedure FormShow(Sender: TObject);

procedure FormKeyDown(Sender: TObject; var Key: Word;

Shift: TShiftState);

procedure ShowHideTimerTimer(Sender: TObject);

procedure dbCbCountry2Click(Sender: TObject);

procedure dbCbCountry3Click(Sender: TObject);

procedure dbCbCountry4Click(Sender: TObject);

procedure cbTypeChange(Sender: TObject);

procedure dbCbZoo3Click(Sender: TObject);

procedure LetKeyPress(Sender: TObject; var Key: Char);

procedure NumKeyPress(Sender: TObject; var Key: Char);

procedure FormHide(Sender: TObject);

private

*{ Private declarations }*

public

*{ Public declarations }*

end;

var

AddFm: TAddFm;

implementation

uses

TimerUnit, AdminMenu, Message;

var

addMenu: TMyMenu;

menuItems:array of TMyMenuItem;

mouseX: integer;

mouseY: integer;

n: integer;

{$R \*.dfm}

procedure AddZoo();

var qryCheck :TQuery;

begin

qryCheck := TQuery.Create(AddFm);

with AddFm do

begin

try

if (Length(lbEdZooName.Text) > 0) and

(Length(lbEdArea.Text) > 0) and

(Length(lbEdCity.Text) > 0) and

(memDescription.Lines.Count > 0) and

(dbCbCountry.KeyValue <> 0) then

begin

qryCheck.Close;

qryCheck.SQL.Clear;

qryCheck.SQL.Add('select \* from DataBase\Zoos where (title = :title)');

qryCheck.ParamByName('title').AsString := lbEdZooName.Text;

qryCheck.ExecSQL;

qryCheck.Open;

if (qryCheck.RecordCount = 0) then

begin

qryAddZoo.Close;

qryAddZoo.ParamByName('title').AsString := lbEdZooName.Text;

qryAddZoo.ParamByName('id\_country').AsInteger := qryShowCountry.FieldByName('id\_country').AsInteger;

qryAddZoo.ParamByName('area').AsInteger := StrToInt(lbEdArea.Text);

qryAddZoo.ParamByName('city').AsString := lbEdCity.Text;

qryAddZoo.ParamByName('ZooDescription').AsMemo := memDescription.Lines.Text;

qryAddZoo.ExecSQL;

MessageFm.mess := 'Зпс дбвеа';

MessageFm.ShowModal;

lbEdZooName.Clear;

lbEdArea.Clear;

lbEdCity.Clear;

memDescription.Clear;

dbCbCountry.KeyValue := 0;

qryShowZoo.Close;

qryShowZoo.Open;

qryShowZoo2.Close;

qryShowZoo2.Open;

qryShowZoo3.Close;

qryShowZoo3.Open;

end

else

begin

MessageFm.mess := 'Ткязпс уеет!';

MessageFm.ShowModal;

end;

end

else

begin

MessageFm.mess := 'Зплеын вепл!';

MessageFm.ShowModal;

end;

finally

qryCheck.Free;

end;

end;

end;

procedure AddMaterial();

var

fileName: string;

qryCheck :TQuery;

begin

qryCheck := TQuery.Create(AddFm);

with AddFm do

begin

try

if (dbCbCountry3.KeyValue <> 0) and

(dbCbAnimal.KeyValue <> 0) and

(dbCbZoo3.KeyValue <> 0) and

(cbType.ItemIndex <> -1) then

begin

qryCheck.SQL.Add('select \* from database\materials');

qryCheck.Close;

qryCheck.ExecSQL;

qryCheck.Open;

qryCheck.Last;

fileName := 'Materials\AnimMaterials\' +

IntToStr(qryShowZoo.FieldByName('id\_zoo').AsInteger) +

IntToStr(qryShowAnim.FieldByName('id\_link').AsInteger) +

IntToStr(qryCheck.FieldByName('id\_material').AsInteger + 1);

qryAddAnimMat.Close;

qryAddAnimMat.ParamByName('id\_link').AsInteger := qryShowAnim.FieldByName('id\_link').AsInteger;

if (cbType.ItemIndex = 0) then

begin

imgAnimal.Picture.SaveToFile(ExtractFilePath(Application.ExeName) +

fileName + '.jpg');

qryAddAnimMat.ParamByName('path').AsString := fileName + '.jpg';

imgAnimal.Picture.Bitmap.FreeImage;

end

else if (cbType.ItemIndex = 1) then

begin

qryAddAnimMat.ParamByName('path').AsString := fileName + '.avi';

CopyFile(PAnsiChar(MP.FileName),

PAnsiChar(ExtractFilePath(Application.ExeName) + fileName + '.avi'),

false);

MP.Close;

MP.FileName := '';

end;

qryAddAnimMat.ParamByName('type').AsInteger := cbType.ItemIndex;

qryAddAnimMat.ExecSQL;

MessageFm.mess := 'Зпс дбвеа';

MessageFm.ShowModal;

dbcbCountry3.KeyValue := 0;

dbCbZoo3.KeyValue := 0;

dbCbAnimal.KeyValue := 0;

cbType.ItemIndex := -1;

imgAnimal.Visible := false;

videoPanel.Visible := false;

end

else

begin

MessageFm.mess := 'Зплеын вепл!';

MessageFm.ShowModal;

end;

finally

qryCheck.Free;

end;

end;

end;

procedure AddSpecies();

var qryCheck: TQuery;

begin

qryCheck := TQuery.Create(AddFm);

with AddFm do

begin

try

if (Length(lbEdSpeciesName.Text) > 0) and

(dbCbHabitat.KeyValue <> 0) and

(dbCbClimate.KeyValue <> 0) and

(dbCbFoodType.KeyValue <>0) then

begin

qryCheck.Close;

qryCheck.SQL.Clear;

qryCheck.SQL.Add('select \* from database\Species where (Name = :name)');

qryCheck.ParamByName('name').AsString := lbEdSpeciesName.Text;

qryCheck.ExecSQL;

qryCheck.Open;

if (qryCheck.RecordCount = 0) then

begin

qryAddSpecies.Close;

qryAddSpecies.ParamByName('name').AsString := lbEdSpeciesName.Text;

qryAddSpecies.ParamByName('id\_habitat').AsInteger := qryShowHabitat.FieldByName('id\_habitat').AsInteger;

qryAddSpecies.ParamByName('id\_climate').AsInteger := qryShowClimate.FieldByName('id\_climate').AsInteger;

qryAddSpecies.ParamByName('id\_foodType').AsInteger := qryShowFoodType.FieldByName('id\_foodType').AsInteger;

qryAddSpecies.ParamByName('Description').AsMemo := memDescriptionSpec.Lines.Text;

qryAddSpecies.ExecSQL;

MessageFm.mess := 'Зпс дбвеа';

MessageFm.ShowModal;

lbEdSpeciesName.Clear;

dbCbHabitat.KeyValue := 0;

dbCbClimate.KeyValue := 0;

dbCbFoodType.KeyValue := 0;

memDescriptionSpec.Clear;

qryShowSpecies.Close;

qryShowSpecies.Open;

end

else

begin

MessageFm.mess := 'Ткязпс уеет!';

MessageFm.ShowModal;

end;

end

else

begin

MessageFm.mess := 'Зплеын вепл!';

MessageFm.ShowModal;

end;

finally

qryCheck.Free;

end;

end;

end;

procedure AddCountry();

var qryCheck: TQuery;

begin

qryCheck := TQuery.Create(AddFm);

with AddFm do

begin

try

if (Length(lbEdCountry.Text) > 0) then

begin

qryCheck.Close;

qryCheck.SQL.Clear;

qryCheck.SQL.Add('select \* from database\countries where (country = :country)');

qryCheck.ParamByName('country').AsString := lbEdCountry.Text;

qryCheck.ExecSQL;

qryCheck.Open;

if (qryCheck.RecordCount = 0) then

begin

qryAddCountry.Close;

qryAddCountry.ParamByName('country').AsString := lbEdCountry.Text;

qryAddCountry.ExecSQL;

MessageFm.mess := 'Зпс дбвеа';

MessageFm.ShowModal;

lbEdCountry.Clear;

qryShowCountry.Close;

qryShowCountry.Open;

qryShowCountry2.Close;

qryShowCountry2.Open;

qryShowCountry3.Close;

qryShowCountry3.Open;

qryShowCountry4.Close;

qryShowCountry4.Open;

end

else

begin

MessageFm.mess := 'Ткязпс уеет!';

MessageFm.ShowModal;

end;

end

else

begin

MessageFm.mess := 'Зплеын вепл!';

MessageFm.ShowModal;

end;

finally

qryCheck.Free;

end;

end;

end;

procedure AddHabitat();

var qryCheck: TQuery;

begin

qryCheck := TQuery.Create(AddFm);

with AddFm do

begin

try

if (Length(lbEdHabitat.Text) > 0) then

begin

qryCheck.Close;

qryCheck.SQL.Clear;

qryCheck.SQL.Add('select \* from database\habitats where (habitat = :habitat)');

qryCheck.ParamByName('habitat').AsString := lbEdHabitat.Text;

qryCheck.ExecSQL;

qryCheck.Open;

if (qryCheck.RecordCount = 0) then

begin

qryAddHabitat.Close;

qryAddHabitat.ParamByName('Habitat').AsString := lbEdHabitat.Text;

qryAddHabitat.ExecSQL;

MessageFm.mess := 'Зпс дбвеа';

MessageFm.ShowModal;

lbEdHabitat.Clear;

qryShowHabitat.Close;

qryShowHabitat.Open;

end

else

begin

MessageFm.mess := 'Ткязпс уеет!';

MessageFm.ShowModal;

end;

end

else

begin

MessageFm.mess := 'Зплеын вепл!';

MessageFm.ShowModal;

end;

finally

qryCheck.Free;

end;

end;

end;

procedure AddFootType();

var qryCheck: TQuery;

begin

qryCheck := TQuery.Create(AddFm);

with AddFm do

begin

try

if (Length(lbEdFoodType.Text) > 0) then

begin

qryCheck.Close;

qryCheck.SQL.Clear;

qryCheck.SQL.Add('select \* from database\FoodTypes where (FoodType = :foodType)');

qryCheck.ParamByName('foodType').AsString := lbEdFoodType.Text;

qryCheck.ExecSQL;

qryCheck.Open;

if (qryCheck.RecordCount = 0) then

begin

qryAddFoodType.Close;

qryAddFoodType.ParamByName('foodType').AsString := lbEdFoodType.Text;

qryAddFoodType.ExecSQL;

MessageFm.mess := 'Зпс дбвеа';

MessageFm.ShowModal;

lbEdFoodType.Clear;

qryShowFoodType.Close;

qryShowFoodType.Open;

end

else

begin

MessageFm.mess := 'Ткязпс уеет!';

MessageFm.ShowModal;

end;

end

else

begin

MessageFm.mess := 'Зплеын вепл!';

MessageFm.ShowModal;

end;

finally

qryCheck.Free;

end;

end;

end;

procedure AddClimate();

var qryCheck: TQuery;

begin

qryCheck := TQuery.Create(AddFm);

with AddFm do

begin

try

if (Length(lbEdClimate.Text) > 0) then

begin

qryCheck.Close;

qryCheck.SQL.Clear;

qryCheck.SQL.Add('select \* from database\Climates where (Climate = :climate)');

qryCheck.ParamByName('climate').AsString := lbEdClimate.Text;

qryCheck.ExecSQL;

qryCheck.Open;

if (qryCheck.RecordCount = 0) then

begin

qryAddClimate.Close;

qryAddClimate.ParamByName('climate').AsString := lbEdClimate.Text;

qryAddClimate.ExecSQL;

MessageFm.mess := 'Зпс дбвеа';

MessageFm.ShowModal;

lbEdClimate.Clear;

qryShowClimate.Close;

qryShowClimate.Open;

end

else

begin

MessageFm.mess := 'Ткязпс уеет!';

MessageFm.ShowModal;

end;

end

else

begin

MessageFm.mess := 'Зплеын вепл!';

MessageFm.ShowModal;

end;

finally

qryCheck.Free;

end;

end;

end;

procedure AddPicToZoo();

var qryCheck: TQuery;

fileName: string;

begin

qryCheck := TQuery.Create(AddFm);

with AddFm do

begin

try

if (dbCbCountry4.KeyValue <> 0) and

(dbCbZoo2.KeyValue <> 0) then

begin

qryCheck.SQL.Add('select \* from database\pictures');

qryCheck.Close;

qryCheck.ExecSQL;

qryCheck.Open;

qryCheck.Last;

fileName := 'Materials\ZooPictures\' +

IntToStr(qryshowZoo2.FieldByName('id\_zoo').AsInteger) +

IntToStr(qryCheck.FieldByName('id\_picture').AsInteger) +

'.jpg';

qryAddZooPic.Close;

qryAddZooPic.ParamByName('id\_zoo').AsInteger := qryShowZoo2.FieldByName('id\_zoo').AsInteger;

qryAddZooPic.ParamByName('Path').AsString := fileName;

qryAddZooPic.ExecSQL;

imgZoo.Picture.SaveToFile(ExtractFilePath(Application.ExeName) +

fileName);

MessageFm.mess := 'Зпс дбвеа';

MessageFm.ShowModal;

dbCbCountry4.KeyValue := 0;

dbCbZoo2.KeyValue := 0;

imgZoo.Picture.Bitmap.FreeImage;

end

else

begin

MessageFm.mess := 'Зплеын вепл!';

MessageFm.ShowModal;

end;

finally

qryCheck.Free;

end;

end;

end;

procedure AddAnimalToZoo();

var qryCheck: TQuery;

begin

qryCheck := TQuery.Create(AddFm);

with AddFm do

begin

try

if (dbCbZoo.KeyValue <> 0) and

(dbCbSpecies.KeyValue <> 0) and

(Length(lbEdAnimName.Text) > 0) then

begin

qryCheck.Close;

qryCheck.SQL.Clear;

qryCheck.SQL.Add('select \* from database\Link where (id\_species = :id\_species and name = :name)');

qryCheck.ParamByName('id\_species').AsInteger := qryShowSpecies.FieldByName('id\_species').AsInteger;

qryCheck.ParamByName('name').AsString := lbEdAnimName.Text;

qryCheck.ExecSQL;

qryCheck.Open;

if (qryCheck.RecordCount = 0) then

begin

qryAddAnimZoo.Close;

qryAddAnimZoo.ParamByName('id\_species').AsInteger := qryShowSpecies.FieldByName('id\_species').AsInteger;

qryAddAnimZoo.ParamByName('id\_zoo').AsInteger := qryShowZoo.FieldByName('id\_zoo').AsInteger;

qryAddAnimZoo.ParamByName('name').AsString := lbEdAnimName.Text;

qryAddAnimZoo.ParamByName('date').AsDate := dtp.Date;

qryAddAnimZoo.ParamByName('note').AsMemo := memNote.Lines.Text;

qryAddAnimZoo.ExecSQL;

MessageFm.mess := 'Зпс дбвеа';

MessageFm.ShowModal;

dbCbCountry.KeyValue := 0;

dbCbSpecies.KeyValue := 0;

lbEdAnimName.Clear;

memNote.Clear;

dtp.DateTime := Now;

dbCbCountry2.KeyValue := 0;

dbCbZoo.KeyValue := 0;

qryShowAnim.Close;

qryShowAnim.Open;

end

else

begin

MessageFm.mess := 'Ткязпс уеет!';

MessageFm.ShowModal;

end;

end

else

begin

MessageFm.mess := 'Зплеын вепл!';

MessageFm.ShowModal;

end;

finally

qryCheck.Free;

end;

end;

end;

procedure LoadPic();

begin

with AddFm do

begin

dlgOpenPic.Execute;

if (Length(dlgOpenPic.FileName) <> 0) then

begin

if (Assigned(imgZoo.Picture.Bitmap)) then

imgZoo.Picture.Bitmap.FreeImage;

imgZoo.Picture.LoadFromFile(dlgOpenPic.FileName);

dlgOpenPic.FileName := '';

end;

end;

end;

procedure LoadPicOrVideo();

begin

with AddFm do

begin

if (cbType.ItemIndex = 0) then

begin

dlgOpenPic.Execute;

if (Length(dlgOpenPic.FileName) <> 0) then

begin

if (Assigned(imgZoo.Picture.Bitmap)) then

imgAnimal.Picture.Bitmap.FreeImage;

imgAnimal.Picture.LoadFromFile(dlgOpenPic.FileName);

dlgOpenPic.FileName := '';

end

else

end

else if (cbType.ItemIndex = 1) then

begin

dlgOpenVideo.Execute;

if (Length(dlgOpenVideo.FileName) <> 0) then

begin

MP.FileName := dlgOpenVideo.FileName;

MP.Open;

MP.Play;

MP.DisplayRect := videoPanel.ClientRect;

dlgOpenVideo.FileName := '';

end;

end

else

begin

MessageFm.mess := 'Н вба тп';

MessageFm.ShowModal;

end;

end;

end;

procedure TAddFm.FormCreate(Sender: TObject);

var i: integer;

begin

n := 11;

addMenu := TMyMenu.Create(n, nil);

SetLength(menuItems, n);

*// Дбвеи зоак*

menuItems[0].name := 'Дбвт';

menuItems[0].x := 218;

menuItems[0].y := 220;

menuItems[0].color := clSilver;

menuItems[0].size := 15;

menuItems[0].onCLick := AddZoo;

*// Дбвеи мтрао*

menuItems[1].name := 'Дбвт';

menuItems[1].x := 542;

menuItems[1].y := 220;

menuItems[1].color := clSilver;

menuItems[1].size := 15;

menuItems[1].onCLick := AddMaterial;

*// Дбвеи вд*

menuItems[2].name := 'Дбвт';

menuItems[2].x := 852;

menuItems[2].y := 166;

menuItems[2].color := clSilver;

menuItems[2].size := 15;

menuItems[2].onCLick := AddSpecies;

*// Дбвеи срн*

menuItems[3].name := 'Дбвт';

menuItems[3].x := 852;

menuItems[3].y := 238;

menuItems[3].color := clSilver;

menuItems[3].size := 15;

menuItems[3].onCLick := AddCountry;

*// Дбвеи сеыоиаи*

menuItems[4].name := 'Дбвт';

menuItems[4].x := 852;

menuItems[4].y := 310;

menuItems[4].color := clSilver;

menuItems[4].size := 15;

menuItems[4].onCLick := AddHabitat;

*// Дбвеи тп птня*

menuItems[5].name := 'Дбвт';

menuItems[5].x := 852;

menuItems[5].y := 382;

menuItems[5].color := clSilver;

menuItems[5].size := 15;

menuItems[5].onCLick := AddFootType;

*// Дбвеи киаа*

menuItems[6].name := 'Дбвт';

menuItems[6].x := 852;

menuItems[6].y := 454;

menuItems[6].color := clSilver;

menuItems[6].size := 15;

menuItems[6].onCLick := AddClimate;

*// Дбвеи крик кзоак*

menuItems[7].name := 'Дбвт';

menuItems[7].x := 542;

menuItems[7].y := 454;

menuItems[7].color := clSilver;

menuItems[7].size := 15;

menuItems[7].onCLick := AddPicToZoo;

menuItems[8].name := 'Дбвт';

menuItems[8].x := 218;

menuItems[8].y := 454;

menuItems[8].color := clSilver;

menuItems[8].size := 15;

menuItems[8].onCLick := AddAnimalToZoo;

*// Згук крик дязоак*

menuItems[9].name := 'Згуиь;

menuItems[9].x := 542;

menuItems[9].y := 400;

menuItems[9].color := clSilver;

menuItems[9].size := 15;

menuItems[9].onCLick := LoadPic;

*// Згук крик лб вдо*

menuItems[10].name := 'Згуиь;

menuItems[10].x := 542;

menuItems[10].y := 180;

menuItems[10].color := clSilver;

menuItems[10].size := 15;

menuItems[10].onCLick := LoadPicOrVideo;

for i := 0 to n - 1 do

begin

menuItems[i].visible := true;

addMenu.Add(menuItems[i]);

end;

AddFm.DoubleBuffered := true;

MenuBox.Invalidate;

end;

procedure TAddFm.MenuBoxPaint(Sender: TObject);

begin

addMenu.DrawMenu(MenuBox.Canvas);

end;

procedure TAddFm.MenuBoxMouseMove(Sender: TObject; Shift: TShiftState; X,

Y: Integer);

begin

mouseX := X;

mouseY := Y;

addMenu.Check(X, Y);

MenuBox.Invalidate;

end;

procedure TAddFm.MenuBoxClick(Sender: TObject);

begin

addMenu.ClickCheck(mouseX, mouseY);

end;

procedure TAddFm.FormShow(Sender: TObject);

begin

lbEdZooName.Clear;

lbEdArea.Clear;

lbEdCity.Clear;

lbEdCountry.Clear;

lbEdHabitat.Clear;

lbEdFoodType.Clear;

lbEdClimate.Clear;

lbEdSpeciesName.Clear;

lbEdAnimName.Clear;

memDescription.Clear;

memNote.Clear;

imgZoo.Picture.Bitmap.FreeImage;

imgAnimal.Picture.Bitmap.FreeImage;

dbCbCountry.KeyValue := 0;

dbCbHabitat.KeyValue := 0;

dbCbFoodType.KeyValue := 0;

dbCbClimate.KeyValue := 0;

dbCbZoo.KeyValue := 0;

dbCbSpecies.KeyValue := 0;

dbCbZoo2.KeyValue := 0;

dbCbAnimal.KeyValue := 0;

dbCbZoo3.KeyValue := 0;

videoPanel.Visible := false;

imgAnimal.Visible := false;

dbCbZoo.Enabled := false;

dbCbZoo2.Enabled := false;

dbCbZoo3.Enabled := false;

dbCbAnimal.Enabled := false;

qryShowCountry.Open;

qryShowCountry2.Open;

qryShowCountry3.Open;

qryShowCountry4.Open;

qryShowSpecies.Open;

qryShowZoo.Open;

qryShowZoo2.Open;

qryShowZoo3.Open;

qryShowAnim.Open;

qryShowHabitat.Open;

qryShowClimate.Open;

qryShowFoodType.Open;

end;

procedure TAddFm.FormKeyDown(Sender: TObject; var Key: Word;

Shift: TShiftState);

begin

if (Key = VK\_ESCAPE) then

ShowHideTimer.Enabled := true;

end;

procedure TAddFm.ShowHideTimerTimer(Sender: TObject);

begin

ShowHide(ShowHideTimer, AddFm, AdminMenuFm);

end;

procedure TAddFm.dbCbCountry2Click(Sender: TObject);

begin

if (dbCbCountry2.KeyValue <> 0) then

begin

dbCbZoo.Enabled := true;

dbCbZoo.KeyValue := 0;

qryShowZoo.Close;

qryShowZoo.ParamByName('id\_country').AsInteger := qryShowCountry2.FieldByName('id\_country').AsInteger;

qryShowZoo.ExecSQL;

qryShowZoo.Open;

end

else

begin

dbCbZoo.Enabled := false;

dbCbZoo.KeyValue := 0;

end;

end;

procedure TAddFm.dbCbCountry3Click(Sender: TObject);

begin

if (dbCbCountry3.KeyValue <> 0) then

begin

dbCbZoo3.Enabled := true;

dbCbZoo3.KeyValue := 0;

qryShowZoo3.Close;

qryShowZoo3.ParamByName('id\_country').AsInteger := qryShowCountry2.FieldByName('id\_country').AsInteger;

qryShowZoo3.ExecSQL;

qryShowZoo3.Open;

end

else

begin

dbCbZoo3.Enabled := false;

dbCbZoo3.KeyValue := 0;

end;

end;

procedure TAddFm.dbCbCountry4Click(Sender: TObject);

begin

if (dbCbCountry4.KeyValue <> 0) then

begin

dbCbZoo2.Enabled := true;

dbCbZoo2.KeyValue := 0;

qryShowZoo2.Close;

qryShowZoo2.ParamByName('id\_country').AsInteger := qryShowCountry2.FieldByName('id\_country').AsInteger;

qryShowZoo2.ExecSQL;

qryShowZoo2.Open;

end

else

begin

dbCbZoo2.Enabled := false;

dbCbZoo2.KeyValue := 0;

end;

end;

procedure TAddFm.cbTypeChange(Sender: TObject);

begin

if (cbType.ItemIndex = 0) then

begin

imgAnimal.Visible := true;

if (MP.FileName <> '') then

MP.Close;

videoPanel.Visible := false;

end

else if (cbType.ItemIndex = 1) then

begin

MP.DisplayRect := videoPanel.ClientRect;

videoPanel.Visible := true;

if (Assigned(imgAnimal.Picture.Bitmap)) then

imgAnimal.Picture.Bitmap.FreeImage;

imgAnimal.Visible := false;

end;

end;

procedure TAddFm.dbCbZoo3Click(Sender: TObject);

begin

if (dbCbZoo3.KeyValue <> 0) then

begin

dbCbAnimal.Enabled := true;

dbCbAnimal.KeyValue := 0;

qryShowAnim.Close;

qryShowAnim.ParamByName('id\_zoo').AsInteger := qryShowZoo3.FieldByName('id\_zoo').AsInteger;

qryShowAnim.ExecSQL;

qryShowAnim.Open;

end

else

begin

dbCbAnimal.Enabled := false;

dbCbAnimal.KeyValue := 0;

end;

end;

procedure TAddFm.LetKeyPress(Sender: TObject; var Key: Char);

begin

if (Key in ['0'..'9']) then

begin

Key := #0;

end;

end;

procedure TAddFm.NumKeyPress(Sender: TObject; var Key: Char);

begin

if (Key in ['0'..'9'] + [#8]) = false then

begin

Key := #0;

end;

end;

procedure TAddFm.FormHide(Sender: TObject);

begin

qryShowCountry.Close;

qryShowCountry2.Close;

qryShowCountry3.Close;

qryShowCountry4.Close;

qryShowSpecies.Close;

qryShowZoo.Close;

qryShowZoo2.Close;

qryShowZoo3.Close;

qryShowAnim.Close;

qryShowHabitat.Close;

qryShowClimate.Close;

qryShowFoodType.Close;

end;

end.

unit AdminMenu;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, StdCtrls, MenuUnit, ExtCtrls;

type

TAdminMenuFm = class(TForm)

bvl: TBevel;

MenuBox: TPaintBox;

ShowHideTimer: TTimer;

procedure FormCreate(Sender: TObject);

procedure MenuBoxPaint(Sender: TObject);

procedure MenuBoxMouseMove(Sender: TObject; Shift: TShiftState; X,

Y: Integer);

procedure MenuBoxClick(Sender: TObject);

procedure ShowHideTimerTimer(Sender: TObject);

private

*{ Private declarations }*

public

*{ Public declarations }*

end;

var

AdminMenuFm: TAdminMenuFm;

admMenu: TMyMenu;

menuItems:array of TMyMenuItem;

n: Integer;

mouseX: integer;

mouseY: integer;

form: TForm;

implementation

uses MainMenu, TimerUnit, Add, Update, Delete;

var count:integer = 0;

{$R \*.dfm}

procedure ToAddMenu();

begin

form := AddFm;

AdminMenuFm.ShowHideTimer.Enabled := true;

end;

procedure ToUpdateMenu();

begin

form := UpdateFm;

AdminMenuFm.ShowHideTimer.Enabled := true;

end;

procedure ToDeleteMenu();

begin

form := DeleteFm;

AdminmenuFm.ShowHideTimer.Enabled := true;

end;

procedure BackToUserMenu();

begin

form := MainMenuFm;

AdminMenuFm.ShowHideTimer.Enabled := true;

end;

procedure TAdminMenuFm.FormCreate(Sender: TObject);

var i, y: integer;

bg: TBitmap;

begin

n := 5;

y := 50;

bg := TBitmap.Create;

bg.LoadFromFile(ExtractFilePath(Application.ExeName) + 'Pictures\BackGround.bmp');

admMenu := TMyMenu.Create(n, bg);

SetLength(menuItems, n);

menuItems[0].name := 'Дбвеи';

menuItems[0].onClick := ToAddMenu;

menuItems[1].name := 'Рдкиоаи';

menuItems[1].onClick := ToUpdateMenu;

menuItems[2].name := 'Уаеи';

menuItems[2].onClick := ToDeleteMenu;

menuItems[3].name := 'Орто;

menuItems[3].onClick := BackToUserMenu;

menuItems[4].name := 'Вхд;

menuItems[4].onClick := Quit;

for i := 0 to n - 1 do

begin

y := y + 50;

menuItems[i].size := 15;

menuItems[i].color := clSilver;

menuItems[i].visible := true;

menuItems[i].x := 100;

menuItems[i].y := menuItems[i].y + y;

admMenu.Add(menuItems[i]);

end;

AdminMenuFm.DoubleBuffered := true;

MenuBox.Invalidate;

end;

procedure TAdminMenuFm.MenuBoxPaint(Sender: TObject);

begin

admMenu.DrawMenu(MenuBox.Canvas);

end;

procedure TAdminMenuFm.MenuBoxMouseMove(Sender: TObject;

Shift: TShiftState; X, Y: Integer);

begin

mouseX := X;

mouseY := Y;

if (admMenu.Check(X, Y)) then

MenuBox.Invalidate;

end;

procedure TAdminMenuFm.MenuBoxClick(Sender: TObject);

begin

admMenu.ClickCheck(mouseX, mouseY);

end;

procedure TAdminMenuFm.ShowHideTimerTimer(Sender: TObject);

begin

ShowHide(ShowHideTimer, AdminMenuFm, form);

end;

end.

unit Delete;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, StdCtrls, ExtCtrls, MenuUnit, TimerUnit;

type

TDeleteFm = class(TForm)

ShowHideTimer: TTimer;

lbl1: TLabel;

bvl1: TBevel;

RadGr1: TRadioGroup;

MenuBox: TPaintBox;

procedure MenuBoxPaint(Sender: TObject);

procedure MenuBoxMouseMove(Sender: TObject; Shift: TShiftState;

X, Y: Integer);

procedure MenuBoxClick(Sender: TObject);

procedure ShowHideTimerTimer(Sender: TObject);

procedure FormCreate(Sender: TObject);

private

*{ Private declarations }*

public

*{ Public declarations }*

end;

var

DeleteFm: TDeleteFm;

implementation

uses

SecondDelete, AdminMenu, Message, FirstDelete;

var mouseX: integer;

mouseY: integer;

menuItems: array of TMyMenuItem;

deleteMenu: TMyMenu;

n: integer;

form: TForm;

{$R \*.dfm}

procedure ToDeleteMenu();

begin

case (DeleteFm.RadGr1.ItemIndex) of

0:begin

FirstDelete.idName := 'id\_country';

FirstDelete.tableName := 'countries';

FirstDelete.fieldName := 'country';

FirstDelete.checkTable := 'Zoos';

FirstDeleteFm.lbl.Caption := 'Вбрт срн:';

form := FirstDeleteFm;

DeleteFm.ShowHideTimer.Enabled := true;

end;

1:begin

FirstDelete.idName := 'id\_foodtype';

FirstDelete.tableName := 'foodtypes';

FirstDelete.fieldName := 'foodtype';

FirstDelete.checkTable := 'Species';

FirstDeleteFm.lbl.Caption := 'Вбрт тпптня';

form := FirstDeleteFm;

DeleteFm.ShowHideTimer.Enabled := true;

end;

2:begin

FirstDelete.idName := 'id\_climate';

FirstDelete.tableName := 'climates';

FirstDelete.fieldName := 'climate';

FirstDelete.checkTable := 'Species';

FirstDeleteFm.lbl.Caption := 'Вбрт киа:';

form := FirstDeleteFm;

DeleteFm.ShowHideTimer.Enabled := true;

end;

3:begin

FirstDelete.idName := 'id\_habitat';

FirstDelete.tableName := 'habitats';

FirstDelete.fieldName := 'habitat';

FirstDelete.checkTable := 'Species';

FirstDeleteFm.lbl.Caption := 'Вбрт сеуоиаи:';

form := FirstDeleteFm;

DeleteFm.ShowHideTimer.Enabled := true;

end;

4:Begin

form := SecondDeleteFm;

DeleteFm.ShowHideTimer.Enabled := true;

end;

else

MessageFm.mess := 'Н вба пнтуаеи!';

MessageFm.ShowModal;

end;

end;

procedure BackToAdminMenu();

begin

form := AdminMenuFm;

DeleteFm.ShowHideTimer.Enabled := true;

end;

procedure TDeleteFm.MenuBoxPaint(Sender: TObject);

begin

deleteMenu.DrawMenu(MenuBox.Canvas);

end;

procedure TDeleteFm.MenuBoxMouseMove(Sender: TObject; Shift: TShiftState;

X, Y: Integer);

begin

mouseX := X;

mouseY := Y;

deleteMenu.Check(X, Y);

MenuBox.Invalidate;

end;

procedure TDeleteFm.MenuBoxClick(Sender: TObject);

begin

deleteMenu.ClickCheck(mouseX, mouseY);

end;

procedure TDeleteFm.ShowHideTimerTimer(Sender: TObject);

begin

ShowHide(ShowHideTimer, self, form);

end;

procedure TDeleteFm.FormCreate(Sender: TObject);

var bg: TBitmap;

begin

n := 2;

SetLength(menuItems, n);

bg := TBitmap.Create;

bg.LoadFromFile(ExtractFilePath(Application.ExeName) + 'Pictures\BackGround2.bmp');

deleteMenu := TMyMenu.Create(n, bg);

menuItems[0].x := 190;

menuItems[0].y := 240;

menuItems[0].name := 'Уаиь;

menuItems[0].size := 15;

menuItems[0].visible := true;

menuItems[0].color := clSilver;

menuItems[0].onClick := ToDeleteMenu;

menuItems[1].x := 190;

menuItems[1].y := 275;

menuItems[1].name := 'Орто;

menuItems[1].size := 15;

menuItems[1].visible := true;

menuItems[1].color := clSilver;

menuItems[1].onClick := BackToAdminMenu;

deleteMenu.Add(menuItems[0]);

deleteMenu.Add(menuItems[1]);

DeleteFm.DoubleBuffered := true;

MenuBox.Invalidate;

end;

end.

unit FirstDelete;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, ExtCtrls, MenuUnit, TimerUnit, DB, DBTables, StdCtrls, DBCtrls;

type

TFirstDeleteFm = class(TForm)

MenuBox: TPaintBox;

ShowHideTimer: TTimer;

dbCbDelete: TDBLookupComboBox;

lbl1: TLabel;

lbl: TLabel;

ds1: TDataSource;

qryShow: TQuery;

procedure MenuBoxPaint(Sender: TObject);

procedure MenuBoxMouseMove(Sender: TObject; Shift: TShiftState;

X, Y: Integer);

procedure MenuBoxClick(Sender: TObject);

procedure ShowHideTimerTimer(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure FormShow(Sender: TObject);

procedure FormHide(Sender: TObject);

private

*{ Private declarations }*

public

*{ Public declarations }*

end;

var

FirstDeleteFm: TFirstDeleteFm;

idName: string;

tableName: string;

fieldName: string;

checkTable: string;

implementation

uses

Delete, Message;

var n: integer;

mouseX: integer;

mouseY: integer;

firstDeleteMenu: TMyMenu;

menuItems: array of TMyMenuItem;

{$R \*.dfm}

procedure DeleteRecord();

var qry: TQuery;

begin

with FirstDeleteFm do

begin

if (dbCbDelete.KeyValue <> 0) then

begin

qry := TQuery.Create(FirstDeleteFm);

try

with qry do

begin

SQL.Clear;

SQL.Add('Select \* from database\' + checkTable);

SQL.Add('where (' + idName + ' = ' +

IntToStr(qryShow.FieldByName(idName).AsInteger) + ')');

Open;

if (RecordCount = 0) then

begin

Close;

SQL.Clear;

SQL.Add('delete from database\' + tableName);

SQL.Add('where (' + idName + ' = ' +

IntToStr(qryShow.FieldByName(idName).AsInteger) + ')');

ExecSQL;

MessageFm.mess := 'Зпс уаеа';

MessageFm.ShowModal;

dbCbDelete.KeyValue := 0;

qryShow.Close;

qryShow.Open;

end

else

begin

MessageFm.mess := 'Уаеи нвзон, ткккэомжтпиет кнршнюцлснсидны!';

MessageFm.ShowModal;

end;

end;

finally

qry.Free;

end;

end

else

begin

MessageFm.mess := 'Зплеын вепл!';

MessageFm.ShowModal;

end;

end;

end;

procedure BackToDeleteMenu();

begin

FirstDeleteFm.ShowHideTimer.Enabled := true;

end;

procedure TFirstDeleteFm.MenuBoxPaint(Sender: TObject);

begin

firstDeleteMenu.DrawMenu(MenuBox.Canvas);

end;

procedure TFirstDeleteFm.MenuBoxMouseMove(Sender: TObject; Shift: TShiftState;

X, Y: Integer);

begin

mouseX := X;

mouseY := Y;

firstDeleteMenu.Check(X, Y);

MenuBox.Invalidate;

end;

procedure TFirstDeleteFm.MenuBoxClick(Sender: TObject);

begin

firstDeleteMenu.ClickCheck(mouseX, mouseY);

end;

procedure TFirstDeleteFm.ShowHideTimerTimer(Sender: TObject);

begin

ShowHide(ShowHideTimer, self, DeleteFm);

end;

procedure TFirstDeleteFm.FormCreate(Sender: TObject);

var bg: TBitmap;

begin

n := 2;

SetLength(menuItems, n);

bg := TBitmap.Create;

bg.LoadFromFile(ExtractFilePath(Application.ExeName) + 'Pictures\BackGround2.bmp');

firstDeleteMenu := TMyMenu.Create(n, bg);

menuItems[0].x := 130;

menuItems[0].y := 110;

menuItems[0].name := 'Уаиь;

menuItems[0].size := 15;

menuItems[0].visible := true;

menuItems[0].color := clSilver;

menuItems[0].onClick := DeleteRecord;

menuItems[1].x := 130;

menuItems[1].y := 140;

menuItems[1].name := 'Орто;

menuItems[1].size := 15;

menuItems[1].visible := true;

menuItems[1].color := clSilver;

menuItems[1].onClick := BackToDeleteMenu;

firstDeleteMenu.Add(menuItems[0]);

firstDeleteMenu.Add(menuItems[1]);

FirstDeleteFm.DoubleBuffered := true;

MenuBox.Invalidate;

end;

procedure TFirstDeleteFm.FormShow(Sender: TObject);

begin

with qryShow do

begin

dbCbDelete.KeyField := idName;

dbCbDelete.ListField := fieldName;

dbCbDelete.KeyValue := 0;

Close;

SQL.Clear;

SQL.Add('Select \* from database\' + tableName);

Open;

end;

end;

procedure TFirstDeleteFm.FormHide(Sender: TObject);

begin

qryShow.Close;

end;

end.

unit FirstUpdate;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, ExtCtrls, StdCtrls, DBCtrls, MenuUnit, TimerUnit, DB, DBTables;

type

TFirstUpdateFm = class(TForm)

MenuBox: TPaintBox;

DBCBShow: TDBLookupComboBox;

LbEdUpdate: TLabeledEdit;

bvl: TBevel;

LbShowUpdate: TLabel;

LbTItle: TLabel;

ShowHideTimer: TTimer;

qryShow: TQuery;

qryUpdate: TQuery;

ds1: TDataSource;

procedure MenuBoxPaint(Sender: TObject);

procedure MenuBoxMouseMove(Sender: TObject; Shift: TShiftState;

X, Y: Integer);

procedure MenuBoxClick(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure ShowHideTimerTimer(Sender: TObject);

procedure FormShow(Sender: TObject);

procedure FormHide(Sender: TObject);

procedure LbEdUpdateKeyPress(Sender: TObject; var Key: Char);

private

*{ Private declarations }*

public

*{ Public declarations }*

end;

var

FirstUpdateFm: TFirstUpdateFm;

fieldName: string;

tableName: string;

idName: string;

implementation

uses

Update, Message;

var firstUpdateMenu: TMyMenu;

menuItems: array of TMyMenuItem;

mouseX: integer;

mouseY: integer;

n: integer;

{$R \*.dfm}

procedure UpdateField();

var qryCheck: TQuery;

begin

qryCheck := TQuery.Create(FirstUpdateFm);

try

if (FirstUpdateFm.DBCBShow.KeyValue <> 0) and

(Length(FirstUpdateFm.LbEdUpdate.Text) > 0) then

begin

qryCheck.SQL.Add('Select \* from database\' + tableName);

qryCheck.SQL.Add('where (' + fieldName + ' = :param)');

qryCheck.Close;

qryCheck.ParamByName('param').AsString := FirstUpdateFm.LbEdUpdate.Text;

qryCheck.Open;

if (qryCheck.RecordCount = 0) then

begin

FirstUpdateFm.qryUpdate.SQL.Clear;

FirstUpdateFm.qryUpdate.SQL.Add('Update database\' + tableName);

FirstUpdateFm.qryUpdate.SQL.Add('set ' + fieldName + ' = :param');

FirstUpdateFm.qryUpdate.SQL.Add('where (' + idName + ' = :param2)');

FirstUpdateFm.qryUpdate.ParamByName('param').AsString :=

FirstUpdateFm.LbEdUpdate.Text;

FirstUpdateFm.qryUpdate.ParamByName('param2').AsInteger :=

FirstUpdateFm.qryShow.FieldByName(idName).AsInteger;

FirstUpdateFm.qryUpdate.ExecSQL;

MessageFm.mess := 'Зпс ордкиоаа';

MessageFm.ShowModal;

FirstUpdateFm.LbEdUpdate.Clear;

FirstUpdateFm.DBCBShow.KeyValue := 0;

FirstUpdateFm.qryShow.Close;

FirstUpdateFm.qryShow.Open;

end

else

begin

MessageFm.mess := 'Ткязпс уеет!';

MessageFm.ShowModal;

FirstUpdateFm.LbEdUpdate.Clear;

FirstUpdateFm.DBCBShow.KeyValue := 0;

end;

end

else

begin

MessageFm.mess := 'Зплеын вепл!';

MessageFm.ShowModal;

end;

finally

qryCheck.Free;

end;

end;

procedure ToUpdateMenu();

begin

FirstUpdateFm.ShowHideTimer.Enabled := true;

end;

procedure TFirstUpdateFm.FormCreate(Sender: TObject);

var bg: TBitmap;

begin

n := 2;

SetLength(menuItems, n);

bg := TBitmap.Create;

bg.LoadFromFile(ExtractFilePath(Application.ExeName) + 'Pictures\BackGround2.bmp');

firstUpdateMenu := TMyMenu.Create(n, bg);

menuItems[0].x := 140;

menuItems[0].y := 180;

menuItems[0].name := 'Рдкиоаь;

menuItems[0].size := 15;

menuItems[0].visible := true;

menuItems[0].color := clSilver;

menuItems[0].onClick := UpdateField;

menuItems[1].x := 170;

menuItems[1].y := 215;

menuItems[1].name := 'Орто;

menuItems[1].size := 15;

menuItems[1].visible := true;

menuItems[1].color := clSilver;

menuItems[1].onClick := ToUpdateMenu;

firstUpdateMenu.Add(menuItems[0]);

firstUpdateMenu.Add(menuItems[1]);

FirstUpdateFm.DoubleBuffered := true;

MenuBox.Invalidate;

end;

procedure TFirstUpdateFm.MenuBoxPaint(Sender: TObject);

begin

firstUpdateMenu.DrawMenu(MenuBox.Canvas);

end;

procedure TFirstUpdateFm.MenuBoxMouseMove(Sender: TObject; Shift: TShiftState;

X, Y: Integer);

begin

mouseX := X;

mouseY := Y;

firstUpdateMenu.Check(X, Y);

MenuBox.Invalidate;

end;

procedure TFirstUpdateFm.MenuBoxClick(Sender: TObject);

begin

firstUpdateMenu.ClickCheck(mouseX, mouseY);

end;

procedure TFirstUpdateFm.ShowHideTimerTimer(Sender: TObject);

begin

ShowHide(ShowHideTimer, FirstUpdateFm, UpdateFm);

end;

procedure TFirstUpdateFm.FormShow(Sender: TObject);

begin

DBCBShow.ListField := fieldName;

DBCBShow.KeyField := idName;

DBCBShow.KeyValue := 0;

qryShow.Close;

qryShow.SQL.Clear;

qryShow.SQL.Add('Select \* from database\' + tableName);

qryShow.ExecSQL;

qryShow.Open;

LbEdUpdate.Clear;

end;

procedure TFirstUpdateFm.FormHide(Sender: TObject);

begin

qryShow.Close;

end;

procedure TFirstUpdateFm.LbEdUpdateKeyPress(Sender: TObject;

var Key: Char);

begin

if (Key in ['0'..'9']) then

begin

Key := #0;

end;

end;

end.

unit MainMenu;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, ExtCtrls, MenuUnit, jpeg, StdCtrls, ShellAPI;

type

TMainMenuFm = class(TForm)

MenuBox: TPaintBox;

ShowHideTimer: TTimer;

lbl1: TLabel;

procedure FormCreate(Sender: TObject);

procedure MenuBoxPaint(Sender: TObject);

procedure MenuBoxMouseMove(Sender: TObject; Shift: TShiftState; X,

Y: Integer);

procedure MenuBoxClick(Sender: TObject);

procedure FormKeyDown(Sender: TObject; var Key: Word;

Shift: TShiftState);

procedure ShowHideTimerTimer(Sender: TObject);

private

*{ Private declarations }*

public

*{ Public declarations }*

end;

var

MainMenuFm: TMainMenuFm;

menuItems:array of TMyMenuItem;

n: integer;

mouseX: integer;

mouseY: integer;

userMenu: TMyMenu;

form: TForm;

implementation

uses

AdminMenu, TimerUnit, Question, ShowZoos, Password;

{$R \*.dfm}

procedure ShowInfo();

begin

form := ShowZoosFm;

MainMenuFm.ShowHideTimer.Enabled := true;

end;

procedure ShowHelp();

begin

ShellExecute(MainMenuFm.Handle, 'open','Help\Help.chm', nil, nil, 1);

end;

procedure TMainMenuFm.FormCreate(Sender: TObject);

var i: integer;

y: integer;

x: integer;

bg: TBitmap;

begin

y := 80;

x := 80;

N := 3;

bg := TBitmap.Create;

bg.LoadFromFile(ExtractFilePath(Application.ExeName) + 'Pictures\BackGround.bmp');

SetLength(menuItems, n);

userMenu := TMyMenu.Create(n, bg);

menuItems[0].name := 'Помт ифраи';

menuItems[0].onClick := ShowInfo;

menuItems[1].name := 'Пмщ';

menuItems[1].onClick := ShowHelp;

menuItems[1].x := 50;

menuItems[2].name := 'Вхд;

menuItems[2].y := 20;

menuItems[2].x := 55;

menuItems[2].onClick := Quit;

for i := 0 to n - 1 do

begin

y := y + 50;

menuItems[i].x := menuItems[i].x + x;

menuItems[i].y := menuItems[i].y + y;

menuItems[i].size := 15;

menuItems[i].color := clSilver;

menuItems[i].visible := True;

userMenu.Add(menuItems[i]);

end;

MainMenuFm.DoubleBuffered := true;

MenuBox.Invalidate;

end;

procedure TMainMenuFm.MenuBoxPaint(Sender: TObject);

begin

userMenu.DrawMenu(MenuBox.Canvas);

end;

procedure TMainMenuFm.MenuBoxMouseMove(Sender: TObject; Shift: TShiftState;

X, Y: Integer);

begin

mouseX := X;

mouseY := Y;

if (userMenu.Check(X, Y)) then

MenuBox.Invalidate;

end;

procedure TMainMenuFm.MenuBoxClick(Sender: TObject);

begin

userMenu.ClickCheck(mouseX, mouseY);

end;

procedure TMainMenuFm.FormKeyDown(Sender: TObject; var Key: Word;

Shift: TShiftState);

begin

if (Key = VK\_F4) then

begin

passwordDlg.ShowModal;

if (passwordDlg.ModalResult = mrYes) then

begin

form := AdminMenuFm;

ShowHideTimer.enabled := True;

end;

end;

end;

procedure TMainMenuFm.ShowHideTimerTimer(Sender: TObject);

begin

ShowHide(ShowHideTimer, MainMenuFm, form);

end;

end.

unit MenuUnit;

interface

uses Graphics, Forms, SysUtils, jpeg, Controls;

type

TClickProc = procedure ();

TMyMenuItem = record

x: integer;

y: integer;

name: string;

size: integer;

color: TColor;

visible: boolean;

onClick: TClickProc;

end;

TMyMenu = class

public

menuItems:array of TMyMenuItem;

constructor Create(max: integer; backGround :TBitmap);

procedure DrawMenu(Canvas: TCanvas);

procedure Add(MenuItem: TMyMenuItem);

function Check(X: integer; Y: Integer): Boolean;

procedure ClickCheck(X: integer; Y: integer);

private

count: integer;

max: integer;

backGround: TBitmap;

end;

procedure Quit();

implementation

uses

Question;

procedure Quit();

begin

QuestionFm.mess := 'В дйтиьохтт вйи';

QuestionFm.ShowModal;

if (QuestionFm.ModalResult = mrYes) then

Application.Terminate;

end;

procedure Draw(Canvas: TCanvas; MenuItem: TMyMenuItem);

begin

Canvas.Font.Size := MenuItem.size;

Canvas.Font.Color := MenuItem.color;

Canvas.TextOut(MenuItem.x, MenuItem.y, MenuItem.name);

end;

function MouseOver(X: integer; Y:integer; MenuItem: TMyMenuItem):TMyMenuItem;

begin

if (MenuItem.x < X) and

(MenuItem.x + Length(MenuItem.name) \* 14 > X) and

(MenuItem.y < Y) and

(MenuItem.y + 20 > Y) then

begin

MenuItem.size := 16;

MenuItem.color := clHotLight;

end

else

begin

MenuItem.size := 15;

MenuItem.color := clSilver;

end;

Result := MenuItem;

end;

constructor TMyMenu.Create(max: integer; backGround: TBitmap);

begin

if (backGround <> nil) then

begin

self.backGround := TBitmap.Create;

self.backGround := backGround;

end;

Self.max := max;

count := 0;

SetLength(MenuItems, max);

end;

procedure TMyMenu.DrawMenu(Canvas: TCanvas);

var i: integer;

begin

Canvas.Brush.Style := bsClear;

if (backGround <> nil) then

Canvas.StretchDraw(Canvas.ClipRect, backGround);

for i := 0 to count - 1 do

begin

if (menuItems[i].visible) then

Draw(Canvas, MenuItems[i]);

end;

end;

procedure TMyMenu.Add(MenuItem: TMyMenuItem);

begin

if (count <> max) then

begin

MenuItems[count] := MenuItem;

count := count + 1;

end;

end;

function TMyMenu.Check(X: integer; Y: Integer): boolean;

var i:integer;

checkResult:boolean;

begin

checkResult := false;

for i := 0 to count - 1 do

begin

MenuItems[i] := MouseOver(X, Y, MenuItems[i]);

checkResult := true;

end;

Result := checkResult;

end;

procedure TMyMenu.ClickCheck(X: integer; Y: integer);

var i: integer;

begin

for i := 0 to count - 1 do

begin

if (MenuItems[i].x < X) and

(MenuItems[i].x + Length(MenuItems[i].name) \* 14 > X) and

(MenuItems[i].y < Y) and

(MenuItems[i].y + 20 > Y) and

(menuItems[i].visible) then

begin

if (Assigned(MenuItems[i].onClick)) then

MenuItems[i].onClick();

end;

end;

end;

end.

unit Message;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, ExtCtrls, MenuUnit, StdCtrls;

type

TMessageFm = class(TForm)

MenuBox: TPaintBox;

messageLab: TLabel;

bvl2: TBevel;

procedure FormShow(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure MenuBoxPaint(Sender: TObject);

procedure MenuBoxMouseMove(Sender: TObject; Shift: TShiftState; X,

Y: Integer);

procedure MenuBoxClick(Sender: TObject);

private

*{ Private declarations }*

public

mess: string;

*{ Public declarations }*

end;

var

MessageFm: TMessageFm;

messMenu: TMyMenu;

mouseX: integer;

mouseY: integer;

implementation

{$R \*.dfm}

procedure AbortMess();

begin

MessageFm.ModalResult := mrAbort;

end;

procedure TMessageFm.FormShow(Sender: TObject);

begin

messageLab.Caption := mess;

MenuBox.Invalidate();

end;

procedure TMessageFm.FormCreate(Sender: TObject);

var menuItem: TMyMenuItem;

bg: TBitmap;

begin

bg := TBitmap.Create;

bg.LoadFromFile(ExtractFilePath(Application.ExeName) + 'Pictures\BackGround2.bmp');

messMenu := TMyMenu.Create(1, bg);

menuItem.x := 165;

menuItem.y := 80;

menuItem.size := 15;

menuItem.name := 'Нзд;

menuItem.visible := True;

menuItem.color := clSilver;

menuItem.onClick := AbortMess;

messMenu.Add(menuItem);

MessageFm.DoubleBuffered := True;

MenuBox.Invalidate;

end;

procedure TMessageFm.MenuBoxPaint(Sender: TObject);

begin

messMenu.DrawMenu(MenuBox.Canvas);

end;

procedure TMessageFm.MenuBoxMouseMove(Sender: TObject; Shift: TShiftState;

X, Y: Integer);

begin

mouseX := X;

mouseY := Y;

if (messMenu.Check(X, Y)) then

MenuBox.Invalidate;

end;

procedure TMessageFm.MenuBoxClick(Sender: TObject);

begin

messMenu.ClickCheck(mouseX, mouseY);

end;

end.

unit Password;

interface

uses Windows, SysUtils, Classes, Graphics, Forms, Controls, StdCtrls,

Buttons, ExtCtrls, MenuUnit;

type

TPasswordDlg = class(TForm)

Label1: TLabel;

Password: TEdit;

MenuBox: TPaintBox;

procedure FormCreate(Sender: TObject);

procedure MenuBoxPaint(Sender: TObject);

procedure MenuBoxMouseMove(Sender: TObject; Shift: TShiftState;

X, Y: Integer);

procedure MenuBoxClick(Sender: TObject);

private

*{ Private declarations }*

public

*{ Public declarations }*

end;

var

PasswordDlg: TPasswordDlg;

implementation

uses

Message;

var mouseX: integer;

mouseY: integer;

menuItems: array of TMyMenuItem;

passwordMenu: TMyMenu;

n: integer;

{$R \*.dfm}

procedure OnYes();

begin

with PasswordDlg do

begin

if (length(Password.Text) > 0) then

begin

if (password.Text = 'hiadmin') then

begin

password.Clear;

ModalResult := mrYes;

end

else

begin

MessageFm.mess := 'Нвры прл';

MessageFm.ShowModal;

password.Clear;

end;

end

else

begin

MessageFm.mess := 'В н веипрл!';

MessageFm.ShowModal;

end;

end;

end;

procedure OnNo();

begin

PasswordDlg.Password.Clear;

PasswordDlg.ModalResult := mrAbort;

end;

procedure TPasswordDlg.FormCreate(Sender: TObject);

var bg: TBitmap;

begin

N := 2;

bg := TBitmap.Create;

bg.LoadFromFile(ExtractFilePath(Application.ExeName) + 'Pictures\BackGround2.bmp');

SetLength(menuItems, n);

passwordMenu := TMyMenu.Create(n, bg);

menuItems[0].name := 'Гтв';

menuItems[0].onClick := OnYes;

menuItems[0].x := 48;

menuItems[0].y := 90;

menuItems[0].size := 15;

menuItems[0].color := clSilver;

menuItems[0].visible := True;

menuItems[1].name := 'Омн';

menuItems[1].onClick := OnNo;

menuItems[1].x := 200;

menuItems[1].y := 90;

menuItems[1].size := 15;

menuItems[1].color := clSilver;

menuItems[1].visible := True;

passwordMenu.Add(menuItems[0]);

passwordMenu.Add(menuItems[1]);

PasswordDlg.DoubleBuffered := true;

MenuBox.Invalidate;

end;

procedure TPasswordDlg.MenuBoxPaint(Sender: TObject);

begin

passwordMenu.DrawMenu(MenuBox.Canvas);

end;

procedure TPasswordDlg.MenuBoxMouseMove(Sender: TObject; Shift: TShiftState;

X, Y: Integer);

begin

mouseX := X;

mouseY := Y;

if (passwordMenu.Check(X, Y)) then

MenuBox.Invalidate;

end;

procedure TPasswordDlg.MenuBoxClick(Sender: TObject);

begin

passwordMenu.ClickCheck(mouseX, mouseY);

end;

end.

unit Question;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, MenuUnit, ExtCtrls, StdCtrls;

type

TQuestionFm = class(TForm)

MenuBox: TPaintBox;

messageLab: TLabel;

bvl1: TBevel;

procedure FormCreate(Sender: TObject);

procedure MenuBoxPaint(Sender: TObject);

procedure MenuBoxMouseMove(Sender: TObject; Shift: TShiftState; X,

Y: Integer);

procedure MenuBoxClick(Sender: TObject);

procedure FormShow(Sender: TObject);

private

*{ Private declarations }*

public

mess: string;

*{ Public declarations }*

end;

var

QuestionFm: TQuestionFm;

questionMenu: TMyMenu;

menuItems:array of TMyMenuItem;

mouseX: integer;

mouseY: integer;

n: integer;

implementation

{$R \*.dfm}

procedure NoClick();

begin

QuestionFm.ModalResult := mrAbort;

*//QuestionFm.Close;*

end;

procedure YesClick();

begin

QuestionFm.ModalResult := mrYes;

*//QuestionFm.Close;*

end;

procedure TQuestionFm.FormCreate(Sender: TObject);

var bg: TBitmap;

begin

n := 2;

bg := TBitmap.Create;

bg.LoadFromFile(ExtractFilePath(Application.ExeName) + 'Pictures\BackGround2.bmp');

questionMenu := TMyMenu.Create(n, bg);

SetLength(menuItems, n);

menuItems[0].x := 125;

menuItems[0].y := 80;

menuItems[0].size := 15;

menuItems[0].color := clSilver;

menuItems[0].name := 'Д';

menuItems[0].visible := true;

menuItems[0].onClick := YesClick;

menuItems[1].x := 225;

menuItems[1].y := 80;

menuItems[1].size := 15;

menuItems[1].color := clSilver;

menuItems[1].name := 'Нт;

menuItems[1].visible := true;

menuItems[1].onClick := NoClick;

questionMenu.Add(menuItems[0]);

questionMenu.Add(menuItems[1]);

QuestionFm.DoubleBuffered := true;

MenuBox.Invalidate;

end;

procedure TQuestionFm.MenuBoxPaint(Sender: TObject);

begin

questionMenu.DrawMenu(MenuBox.Canvas);

end;

procedure TQuestionFm.MenuBoxMouseMove(Sender: TObject; Shift: TShiftState;

X, Y: Integer);

begin

mouseX := X;

mouseY := Y;

if (questionMenu.Check(X, Y)) then

MenuBox.Invalidate;

end;

procedure TQuestionFm.MenuBoxClick(Sender: TObject);

begin

questionMenu.ClickCheck(mouseX, mouseY);

end;

procedure TQuestionFm.FormShow(Sender: TObject);

begin

messageLab.Caption := mess;

MenuBox.Invalidate;

end;

end.

unit ScreenSaver;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, ExtCtrls, StdCtrls, XPMan, jpeg;

type

TScreenSaverFm = class(TForm)

Timer: TTimer;

bvl: TBevel;

BackGround: TImage;

XpManifest: TXPManifest;

ShowHideTimer: TTimer;

procedure FormCreate(Sender: TObject);

procedure TimerTimer(Sender: TObject);

procedure ShowHideTimerTimer(Sender: TObject);

private

*{ Private declarations }*

public

*{ Public declarations }*

end;

var

ScreenSaverFm: TScreenSaverFm;

timeElepsed: integer;

implementation

uses

MainMenu, TimerUnit;

{$R \*.dfm}

procedure TScreenSaverFm.FormCreate(Sender: TObject);

begin

ScreenSaverFm.DoubleBuffered := true;

end;

procedure TScreenSaverFm.TimerTimer(Sender: TObject);

begin

Timer.Enabled := false;

ScreenSaverFm.Enabled := false;

ShowHideTimer.Enabled := true;

end;

procedure TScreenSaverFm.ShowHideTimerTimer(Sender: TObject);

begin

ShowHide(ShowHideTimer, ScreenSaverFm, MainMenuFm);

end;

end.

unit Search;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, ExtCtrls, MenuUnit, StdCtrls;

type

TSearchFm = class(TForm)

MenuBox: TPaintBox;

HideTimer: TTimer;

GroupBox: TGroupBox;

chk1: TCheckBox;

chk2: TCheckBox;

chk3: TCheckBox;

lbEdZooName: TLabeledEdit;

lbEdCity: TLabeledEdit;

lbEdSpecies: TLabeledEdit;

lblLbTItle: TLabel;

procedure MenuBoxPaint(Sender: TObject);

procedure MenuBoxMouseMove(Sender: TObject; Shift: TShiftState;

X, Y: Integer);

procedure MenuBoxClick(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure HideTimerTimer(Sender: TObject);

procedure FormShow(Sender: TObject);

procedure chk1Click(Sender: TObject);

procedure chk2Click(Sender: TObject);

procedure chk3Click(Sender: TObject);

private

*{ Private declarations }*

public

*{ Public declarations }*

end;

var

SearchFm: TSearchFm;

procedure ClearSearchResult();

implementation

uses

ShowZoos, Message, TimerUnit;

var mouseX: integer;

mouseY: integer;

n: integer;

searchMenu: TMyMenu;

menuItems: array of TMyMenuItem;

{$R \*.dfm}

procedure UseSearch();

var condition1: string;

condition2: string;

begin

with SearchFm do

begin

if (not chk1.Checked) and

(not chk2.Checked) and

(not chk3.Checked) then

begin

MessageFm.mess := 'Зплеын вепл!';

MessageFm.ShowModal;

end

else

begin

ClearSearchResult();

if ((chk1.Checked) or (chk2.Checked)) then

begin

with ShowZoosFm do

begin

if (Length(lbEdZooName.Text) > 0) then

begin

condition1 := 'z.Title like "%' + lbEdZooName.Text + '%"';

end;

if (Length(lbEdCity.Text) > 0) then

begin

condition2 := 'z.City like "%' + lbEdCity.Text + '%"';

end;

qryShowZoo.Close;

qryShowZoo.SQL.Clear;

qryShowZoo.SQL.Add('Select \* from database\Zoos z');

if (Length(condition1) > 0) and (Length(condition2) > 0) then

begin

qryShowZoo.SQL.Add('Where ' + condition1 + ' and ' + condition2 + '');

end

else if (Length(condition1) > 0) then

begin

qryShowZoo.SQL.Add('Where ' + condition1 +'')

end

else

begin

qryShowZoo.SQL.Add('Where ' + condition2 +'')

end;

qryShowZoo.Open;

qryShowCountry.SQL.Clear;

qryShowCountry.SQL.Add('select \* from database\countries c');

qryShowCountry.SQL.Add('where (c.id\_country in (select id\_country from database\zoos z');

qryShowCountry.SQL.Add('Where (z.Title like "%' + lbEdZooName.Text + '%") group by id\_country))');

qryShowCountry.Open;

end;

end;

if (chk3.Checked) then

begin

with ShowZoosFm do

begin

if (Length(lbedSpecies.Text) > 0) then

begin

qryShowAnim.Close;

qryShowAnim.SQL.Clear;

qryShowAnim.SQL.Add('select \* from database\link l');

qryShowAnim.SQL.Add('inner join database\species s');

qryShowAnim.SQL.Add('on l.id\_species = s.id\_species');

qryShowAnim.SQL.Add('inner join database\habitats h');

qryShowAnim.SQL.Add('on s.id\_habitat = h.id\_habitat');

qryShowAnim.SQL.Add('inner join database\FoodTypes f');

qryShowAnim.SQL.Add('on s.id\_foodtype = f.id\_foodtype');

qryShowAnim.SQL.Add('inner join database\Climates c');

qryShowAnim.SQL.Add('on s.id\_climate = c.id\_climate');

qryShowAnim.SQL.Add('where (s.Name like "' + lbEdSpecies.Text + '%" or l.id\_zoo = :id\_zoo)');

qryShowAnim.ParamByName('id\_zoo').AsInteger := 0;

qryShowAnim.Open;

if (chk1.Checked) or (chk2.Checked) then

begin

qryShowZoo.Close;

qryShowZoo.SQL.Add('and z.id\_zoo in (select id\_zoo from database\link l');

qryShowZoo.SQL.Add('where (l.id\_species in (select id\_species from database\species s');

qryShowZoo.SQL.Add('where (s.Name like "' + lbEdSpecies.Text + '%"))))');

qryShowZoo.Open;

end

else

begin

qryShowZoo.Close;

qryShowZoo.SQL.Clear;

qryShowZoo.SQL.Add('Select \* from database\Zoos z');

qryShowZoo.SQL.Add('where (z.id\_zoo in (select id\_zoo from database\link l');

qryShowZoo.SQL.Add('where (l.id\_species in (select id\_species from database\species s');

qryShowZoo.SQL.Add('where (s.Name like "' + lbEdSpecies.Text + '%")))) or z.id\_country = :id\_country)');

qryShowZoo.ParamByName('id\_country').AsInteger := 0;

qryShowZoo.Open;

end;

qryShowCountry.SQL.Clear;

qryShowCountry.SQL.Add('select \* from database\countries c');

qryShowCountry.SQL.Add('where (c.id\_country in (');

qryShowCountry.SQL.Add('Select id\_country from database\Zoos z');

qryShowCountry.SQL.Add('where (z.id\_zoo in (select id\_zoo from database\link l');

qryShowCountry.SQL.Add('where (l.id\_species in (select id\_species from database\species s');

qryShowCountry.SQL.Add('where (s.Name like "' + lbEdSpecies.Text + '%")))))');

qryShowCountry.SQL.Add('group by id\_country))');

qryShowCountry.Open;

end;

end;

end;

SearchFm.HideTimer.Enabled := true;

end;

end;

end;

procedure ClearSearchResult();

begin

with SearchFm do

begin

chk1.Checked := false;

chk2.Checked := false;

chk3.Checked := false;

end;

with ShowZoosFm do

begin

qryShowCountry.Close;

qryShowCountry.SQL.Clear;

qryShowCountry.SQL.Add('select \* from database\countries');

qryShowCountry.Open;

qryShowZoo.Close;

qryShowZoo.SQL.Clear;

qryShowZoo.SQL.Add('select \* from database\zoos z');

qryShowZoo.SQL.Add('where (id\_Country = :id\_country)');

qryShowZoo.ParamByName('id\_country').AsInteger := 0;

qryShowZoo.Open;

qryShowAnim.Close;

qryShowAnim.SQL.Clear;

qryShowAnim.SQL.Add('select \* from database\link l');

qryShowAnim.SQL.Add('inner join database\species s');

qryShowAnim.SQL.Add('on l.id\_species = s.id\_species');

qryShowAnim.SQL.Add('inner join database\habitats h');

qryShowAnim.SQL.Add('on s.id\_habitat = h.id\_habitat');

qryShowAnim.SQL.Add('inner join database\FoodTypes f');

qryShowAnim.SQL.Add('on s.id\_foodtype = f.id\_foodtype');

qryShowAnim.SQL.Add('inner join database\Climates c');

qryShowAnim.SQL.Add('on s.id\_climate = c.id\_climate');

qryShowAnim.SQL.Add('where (l.id\_zoo = :id\_zoo)');

qryShowAnim.ParamByName('id\_zoo').AsInteger := 0;

qryShowAnim.Open;

qryShowCountry.First;

dbCbCountry.KeyValue := qryShowCountry.FieldByName('id\_country').AsInteger;

dbcbCountryClick(ShowZOosFm);

dbGrdZooCellClick(dbGrdZoo.Columns.Items[0]);

dbGrdAnimCellClick(dbGrdAnim.Columns.Items[0]);

end;

end;

procedure HideSearch();

begin

SearchFm.HideTimer.Enabled := true;

end;

procedure TSearchFm.MenuBoxPaint(Sender: TObject);

begin

searchMenu.DrawMenu(MenuBox.Canvas);

end;

procedure TSearchFm.MenuBoxMouseMove(Sender: TObject; Shift: TShiftState;

X, Y: Integer);

begin

mouseX := X;

mouseY := Y;

searchMenu.Check(X, Y);

MenuBox.Invalidate;

end;

procedure TSearchFm.MenuBoxClick(Sender: TObject);

begin

searchMenu.ClickCheck(mouseX, mouseY);

end;

procedure TSearchFm.FormCreate(Sender: TObject);

var bg: TBitmap;

begin

n := 3;

SetLength(menuItems, n);

bg := TBitmap.Create();

bg.LoadFromFile(ExtractFilePath(Application.ExeName) + 'Pictures\BackGround2.bmp');

searchMenu := TMyMenu.Create(n, bg);

menuItems[0].x := 55;

menuItems[0].y := 265;

menuItems[0].size := 15;

menuItems[0].color := clSilver;

menuItems[0].name := 'Нйи;

menuItems[0].visible := true;

menuItems[0].onClick := UseSearch;

menuItems[1].x := 200;

menuItems[1].y := 265;

menuItems[1].size := 15;

menuItems[1].color := clSilver;

menuItems[1].name := 'Срст';

menuItems[1].visible := true;

menuItems[1].onClick := ClearSearchResult;

menuItems[2].x := 365;

menuItems[2].y := 265;

menuItems[2].size := 15;

menuItems[2].color := clSilver;

menuItems[2].name := 'Срт';

menuItems[2].visible := true;

menuItems[2].onClick := HideSearch;

searchMenu.Add(menuItems[0]);

searchMenu.Add(menuItems[1]);

searchMenu.Add(menuItems[2]);

Self.DoubleBuffered := true;

MenuBox.Invalidate;

end;

procedure TSearchFm.HideTimerTimer(Sender: TObject);

begin

if (self.AlphaBlendValue <> 0) then

begin

if (self.AlphaBlendValue = 255) then

begin

TimerUnit.BlockInput(true);

end;

self.AlphaBlendValue := self.AlphaBlendValue - 5;

end

else

begin

HideTimer.Enabled := false;

TimerUnit.BlockInput(false);

self.Hide;

end;

end;

procedure TSearchFm.FormShow(Sender: TObject);

begin

chk1.Checked := false;

chk2.Checked := false;

chk3.Checked := false;

lbEdZooName.Clear;

lbEdCity.Clear;

lbEdSpecies.Clear;

end;

procedure TSearchFm.chk1Click(Sender: TObject);

begin

lbEdZooName.Enabled := chk1.Checked;

end;

procedure TSearchFm.chk2Click(Sender: TObject);

begin

lbEdCity.Enabled := chk2.Checked;

end;

procedure TSearchFm.chk3Click(Sender: TObject);

begin

lbEdSpecies.Enabled := chk3.Checked;

end;

end.

unit SecondDelete;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, ExtCtrls, DB, DBTables, DBCtrls, StdCtrls, MenuUnit, TimerUnit,

jpeg, MPlayer;

type

TSecondDeleteFm = class(TForm)

MenuBox: TPaintBox;

bvl4: TBevel;

lbl7: TLabel;

lbl9: TLabel;

dbCbZoo3: TDBLookupComboBox;

lbl10: TLabel;

imgZoo: TImage;

bvl5: TBevel;

lbl11: TLabel;

lbl12: TLabel;

dbCbZoo4: TDBLookupComboBox;

lbl13: TLabel;

imgAnim: TImage;

videoPanel: TPanel;

dbCbAnim2: TDBLookupComboBox;

lbl14: TLabel;

bvl3: TBevel;

dbCbAnim: TDBLookupComboBox;

lbl6: TLabel;

dbCbZoo2: TDBLookupComboBox;

lbl5: TLabel;

lbl4: TLabel;

dbCbZoo: TDBLookupComboBox;

lbl3: TLabel;

lbl2: TLabel;

bvl2: TBevel;

ds1: TDataSource;

ds2: TDataSource;

ds3: TDataSource;

ds4: TDataSource;

ds5: TDataSource;

ds6: TDataSource;

qryShowZoo: TQuery;

qryShowZoo2: TQuery;

qryShowZoo3: TQuery;

qryShowAnim: TQuery;

qryShowZoo4: TQuery;

qryShowAnim2: TQuery;

qryShowPic: TQuery;

qryShowMat: TQuery;

ShowHideTimer: TTimer;

imgBackGround: TImage;

bvl1: TBevel;

dbCbSpecies: TDBLookupComboBox;

lbl1: TLabel;

lbl8: TLabel;

ds7: TDataSource;

qryShowSpecies: TQuery;

MP: TMediaPlayer;

procedure FormShow(Sender: TObject);

procedure FormHide(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure MenuBoxPaint(Sender: TObject);

procedure MenuBoxMouseMove(Sender: TObject; Shift: TShiftState;

X, Y: Integer);

procedure MenuBoxClick(Sender: TObject);

procedure ShowHideTimerTimer(Sender: TObject);

procedure dbCbZoo3Click(Sender: TObject);

procedure dbCbZoo2Click(Sender: TObject);

procedure dbCbZoo4Click(Sender: TObject);

procedure dbCbAnim2Click(Sender: TObject);

private

*{ Private declarations }*

public

*{ Public declarations }*

end;

var

SecondDeleteFm: TSecondDeleteFm;

implementation

uses

Delete, Question, Message;

var mouseX: integer;

mouseY: integer;

menuItems: array of TMyMenuItem;

secondDeleteMenu: TMyMenu;

n: integer;

dir: string;

currPic: integer;

currMat: integer;

{$R \*.dfm}

procedure BackToDeleteMenu();

begin

SecondDeleteFm.ShowHideTimer.Enabled := true;

end;

procedure DeleteZoo();

var qry: TQuery;

id: integer;

i: integer;

begin

with SecondDeleteFm do

begin

if (dbCbZoo.KeyValue <> 0) then

begin

QuestionFm.mess := 'В дйтиьохтт вйи';

QuestionFm.ShowModal;

if (QuestionFm.ModalResult = mrYes) then

begin

id := qryShowZoo.FieldByName('id\_Zoo').AsInteger;

qry := TQuery.Create(SecondDeleteFm);

try

with qry do

begin

Close;

SQL.Add('Delete from database\Zoos');

SQL.Add('where (id\_zoo = ' + IntToStr(id) + ')');

ExecSQL;

SQL.Clear;

SQL.Add('Select \* from database\pictures');

SQL.Add('where (id\_zoo = ' + IntToStr(id) + ')');

Open;

First;

for i := 1 to RecordCount do

Begin

DeleteFile(dir + FieldByName('Path').AsString);

Next;

end;

SQL.Clear;

SQL.Add('Delete from database\Pictures');

SQL.Add('where (id\_Zoo = ' + IntToStr(id) + ')');

ExecSQL;

SQL.Clear;

SQL.Add('Select \* from database\Materials');

SQL.Add('where (id\_link in (select id\_link from');

SQL.Add('database\link where (id\_zoo = ' + IntToStr(id) +')))');

Open;

First;

for i := 1 to RecordCount do

Begin

DeleteFile(dir + FieldByName('Path').AsString);

Next;

end;

SQL.Clear;

SQL.Add('Delete from database\Materials');

SQL.Add('where (id\_link in (select id\_link from');

SQL.Add('database\link where (id\_zoo = ' + IntToStr(id) +')))');

ExecSQL;

SQL.Clear;

SQL.Add('Delete from database\link');

SQL.Add('where (id\_zoo = ' + IntToStr(id) + ')');

ExecSQL;

MessageFm.mess := 'Зпс уаеа';

MessageFm.ShowModal;

dbcbZoo.KeyValue := 0;

qryShowZoo.Close;

qryShowZoo.Open;

qryShowZoo2.Close;

qryShowZoo2.Open;

qryShowZoo3.Close;

qryShowZoo3.Open;

qryShowZoo4.Close;

qryShowZoo4.Open;

end;

finally

qry.Free;

end;

end;

end

else

begin

MessageFm.mess := 'Зплеын вепл!';

MessageFm.ShowModal;

end;

end;

end;

procedure DeleteAnimal();

var qry: TQuery;

id: integer;

i: integer;

begin

with SecondDeleteFm do

begin

if (dbcbZoo2.KeyValue <> 0) and (dbcbAnim.KeyValue <> 0) then

begin

QuestionFm.mess := 'В дйтиьохтт вйи';

QuestionFm.ShowModal;

if (QuestionFm.ModalResult = mrYes) then

begin

id := qryShowAnim2.FieldByName('id\_link').AsInteger;

qry := TQuery.Create(SecondDeleteFm);

try

with qry do

begin

Close;

SQL.Add('delete from database\link');

SQL.Add('where (id\_link = ' + IntToStr(id) + ')');

ExecSQL;

SQL.Clear;

SQL.Add('Select \* from database\Materials');

SQL.Add('where (id\_link = ' + IntToStr(id) + ')');

Open;

First;

for i := 1 to RecordCount do

Begin

DeleteFile(dir + FieldByName('Path').AsString);

Next;

end;

SQL.Clear;

SQL.Add('delete from database\materials');

SQL.Add('where (id\_link = ' + IntToStr(id) + ')');

ExecSQL;

MessageFm.mess := 'Зпс уаеа';

MessageFm.ShowModal;

qryShowAnim.Close;

qryShowAnim.Open;

qryShowAnim2.Close;

qryShowAnim2.Open;

dbCbAnim2.KeyValue := 0;

dbCbAnim.KeyValue := 0;

end;

finally

qry.Free;

end;

end;

end

else

begin

MessageFm.mess := 'Зплеын вепл!';

MessageFm.ShowModal;

end;

end;

end;

procedure DeletePhotos();

var qry: TQuery;

begin

with SecondDeleteFM do

begin

if (dbCbZoo3.KeyValue <> 0) then

begin

if (qryShowPic.RecordCount <> 0) then

begin

QuestionFm.mess := 'В дйтиьохтт вйи';

QuestionFm.ShowModal;

if (QuestionFm.ModalResult = mrYes) then

begin

qry := TQuery.Create(SecondDeleteFm);

try

with qry do

begin

DeleteFile(dir + qryShowPic.FieldByName('Path').AsString);

qry.Close;

qry.SQL.Add('delete from database\pictures');

qry.SQL.Add('where (id\_picture = ' +

IntToStr(qryShowPic.FieldByName('id\_picture').AsInteger) + ')');

qry.ExecSQL;

qryShowPic.Close;

qryShowPic.Open;

dbCbZoo3.KeyValue := 0;

qryShowPic.First;

imgZoo.Picture.Bitmap.FreeImage;

MessageFm.mess := 'Зпс уаеа';

MessageFm.ShowModal;

secondDeleteMenu.menuItems[6].visible := false;

secondDeleteMenu.menuItems[7].visible := false;

end;

finally

qry.Free;

end;

end;

end

else

begin

MessageFm.mess := 'Кдноузоак нт крио!';

MessageFm.ShowModal;

end;

end

else

begin

MessageFm.mess := 'Зплеын вепл!';

MessageFm.ShowModal;

end;

end;

end;

procedure DeleteMaterials();

var qry: TQuery;

begin

with SecondDeleteFM do

begin

if (dbCbZoo4.KeyValue <> 0) and (dbCbAnim2.KeyValue <> 0) then

begin

if (qryShowMat.RecordCount <> 0) then

begin

QuestionFm.mess := 'В дйтиьохтт вйи';

QuestionFm.ShowModal;

if (QuestionFm.ModalResult = mrYes) then

begin

qry := TQuery.Create(SecondDeleteFm);

try

with qry do

begin

DeleteFile(dir + qryShowMat.FieldByName('Path').AsString);

qry.Close;

qry.SQL.Add('delete from database\Materials');

qry.SQL.Add('where (id\_material = ' +

IntToStr(qryShowMat.FieldByName('id\_material').AsInteger) + ')');

qry.ExecSQL;

qryShowMat.Close;

qryShowMat.Open;

dbCbZoo4.KeyValue := 0;

dbCbAnim2.KeyValue := 0;

qryShowMat.First;

imgAnim.Picture.Bitmap.FreeImage;

MP.Close;

MP.FileName := '';

MessageFm.mess := 'Зпс уаеа';

MessageFm.ShowModal;

secondDeleteMenu.menuItems[8].visible := false;

secondDeleteMenu.menuItems[9].visible := false;

end;

finally

qry.Free;

end;

end;

end

else

begin

MessageFm.mess := 'Кдноузоак нт крио!';

MessageFm.ShowModal;

end;

end

else

begin

MessageFm.mess := 'Зплеын вепл!';

MessageFm.ShowModal;

end;

end;

end;

procedure DeleteSpecies();

var qry: TQuery;

id: integer;

begin

with SecondDeleteFm do

begin

if (dbcbSpecies.KeyValue <> 0) then

begin

QuestionFm.mess := 'В дйтиьохтт вйи';

QuestionFm.ShowModal;

if (QuestionFm.ModalResult = mrYes) then

begin

id := qryShowSpecies.FieldByName('id\_species').AsInteger;

qry := TQuery.Create(SecondDeleteFm);

try

with qry do

begin

Close;

SQL.Add('select \* from database\link');

SQL.Add('where (id\_species = ' + IntToStr(id) + ')');

Open;

if (RecordCount = 0) then

begin

SQL.Clear;

SQL.Add('delete from database\species');

SQL.Add('where (id\_species = ' + IntToStr(id) + ')');

ExecSQL;

MessageFm.mess := 'Зпс уаеа';

MessageFm.ShowModal;

qryShowSpecies.Close;

qryShowSpecies.Open;

dbCbSpecies.KeyValue := 0;

end

else

begin

MessageFm.mess := 'Уаеи нвзон, ткккэомжтпиет кнршнюцлснсидны!';

MessageFm.ShowModal;

end;

end;

finally

qry.Free;

end;

end;

end

else

begin

MessageFm.mess := 'Зплеын вепл!';

MessageFm.ShowModal;

end;

end;

end;

procedure PriorZooPic();

begin

with SecondDeleteFm do

begin

if (currPic <> 0) then

begin

qryShowPic.Prior;

Dec(currPic);

imgZoo.Picture.LoadFromFile(dir + qryShowPic.FieldByName('Path').AsString);

end;

end;

end;

procedure NextZooPic();

begin

with SecondDeleteFm do

begin

if (currPic <> qryShowPic.RecordCount) then

begin

qryShowPic.Next;

inc(currPic);

imgZoo.Picture.LoadFromFile(dir + qryShowPic.FieldByName('Path').AsString);

end;

end;

end;

procedure PriorMaterial();

begin

with SecondDeleteFm do

begin

if (currMat <> 0) then

begin

qryShowMat.Prior;

Dec(currMat);

if (qryShowMat.FieldByName('type').AsInteger = 0) then

begin

imgAnim.Show;

videoPanel.Hide;

imgAnim.Picture.LoadFromFile(dir + qryShowMat.FieldByName('Path').AsString);

MP.Close;

MP.FileName := '';

end

else

begin

imgAnim.Hide;

videoPanel.Show;

imgAnim.Picture.Bitmap.FreeImage;

MP.FileName := dir + qryShowMat.FieldByName('Path').AsString;

MP.Open;

MP.Play;

MP.DisplayRect := videoPanel.ClientRect;

end;

end;

end;

end;

procedure NextMaterial();

begin

with SecondDeleteFm do

begin

if (currMat <> qryShowMat.RecordCount) then

begin

qryShowMat.Next;

inc(currMat);

if (qryShowMat.FieldByName('type').AsInteger = 0) then

begin

imgAnim.Show;

videoPanel.Hide;

imgAnim.Picture.LoadFromFile(dir + qryShowMat.FieldByName('Path').AsString);

MP.Close;

MP.FileName := '';

end

else

begin

imgAnim.Hide;

videoPanel.Show;

imgAnim.Picture.Bitmap.FreeImage;

MP.FileName := dir + qryShowMat.FieldByName('Path').AsString;

MP.Open;

MP.Play;

MP.DisplayRect := videoPanel.ClientRect;

end;

end;

end;

end;

procedure TSecondDeleteFm.FormShow(Sender: TObject);

begin

qryShowZoo.Open;

qryShowZoo2.Open;

qryShowZoo3.Open;

qryShowZoo4.Open;

qryShowAnim.Open;

qryShowAnim2.Open;

qryShowSpecies.Open;

dbCbZoo3.KeyValue := 0;

dbCbZoo4.KeyValue := 0;

dbCbAnim2.KeyValue := 0;

dbCbAnim.KeyValue := 0;

dbCbZoo2.KeyValue := 0;

dbCbZoo.KeyValue := 0;

dbCbSpecies.KeyValue := 0;

imgZoo.Picture.Bitmap.FreeImage;

imgAnim.Picture.Bitmap.FreeImage;

MP.Close;

MP.FileName := '';

end;

procedure TSecondDeleteFm.FormHide(Sender: TObject);

begin

qryShowZoo.Close;

qryShowZoo2.Close;

qryShowZoo3.Close;

qryShowZoo4.Close;

qryShowAnim.Close;

qryShowAnim2.Close;

qryShowSpecies.Close;

end;

procedure TSecondDeleteFm.FormCreate(Sender: TObject);

var i: integer;

begin

n := 10;

dir := ExtractFilePath(Application.ExeName);

secondDeleteMenu := TMyMenu.Create(n, nil);

SetLength(menuItems, n);

*// Орто*

menuItems[0].x := 740;

menuItems[0].y := 510;

menuItems[0].size := 15;

menuItems[0].color := clSilver;

menuItems[0].name := 'Орто;

menuItems[0].visible := true;

menuItems[0].onClick := BackToDeleteMenu;

*// Уаеи зоак*

menuItems[1].x := 190;

menuItems[1].y := 165;

menuItems[1].size := 15;

menuItems[1].color := clSilver;

menuItems[1].name := 'Уаиь;

menuItems[1].visible := true;

menuItems[1].onClick := DeleteZoo;

*// Уаеи жвтоо*

menuItems[2].x := 190;

menuItems[2].y := 330;

menuItems[2].size := 15;

menuItems[2].color := clSilver;

menuItems[2].name := 'Уаиь;

menuItems[2].visible := true;

menuItems[2].onClick := DeleteAnimal;

*// Уаеи фтгаи*

menuItems[3].x := 470;

menuItems[3].y := 460;

menuItems[3].size := 15;

menuItems[3].color := clSilver;

menuItems[3].name := 'Уаиь;

menuItems[3].visible := true;

menuItems[3].onClick := DeletePhotos;

*// Уаеи мтрао*

menuItems[4].x := 740;

menuItems[4].y := 460;

menuItems[4].size := 15;

menuItems[4].color := clSilver;

menuItems[4].name := 'Уаиь;

menuItems[4].visible := true;

menuItems[4].onClick := DeleteMaterials;

*// Уаеи вдв*

menuItems[5].x := 190;

menuItems[5].y := 460;

menuItems[5].size := 15;

menuItems[5].color := clSilver;

menuItems[5].name := 'Уаиь;

menuItems[5].visible := true;

menuItems[5].onClick := DeleteSpecies;

menuItems[6].x := 300;

menuItems[6].y := 424;

menuItems[6].size := 15;

menuItems[6].color := clSilver;

menuItems[6].name := '<';

menuItems[6].visible := false;

menuItems[6].onClick := PriorZooPic;

menuItems[7].x := 535;

menuItems[7].y := 424;

menuItems[7].size := 15;

menuItems[7].color := clSilver;

menuItems[7].name := '>';

menuItems[7].visible := false;

menuItems[7].onClick := NextZooPic;

menuItems[8].x := 580;

menuItems[8].y := 424;

menuItems[8].size := 15;

menuItems[8].color := clSilver;

menuItems[8].name := '<';

menuItems[8].visible := false;

menuItems[8].onClick := PriorMaterial;

menuItems[9].x := 805;

menuItems[9].y := 424;

menuItems[9].size := 15;

menuItems[9].color := clSilver;

menuItems[9].name := '>';

menuItems[9].visible := false;

menuItems[9].onClick := NextMaterial;

for i := 0 to n - 1 do

secondDeleteMenu.Add(menuItems[i]);

self.DoubleBuffered := true;

MenuBox.Invalidate;

end;

procedure TSecondDeleteFm.MenuBoxPaint(Sender: TObject);

begin

secondDeleteMenu.DrawMenu(MenuBox.Canvas);

end;

procedure TSecondDeleteFm.MenuBoxMouseMove(Sender: TObject; Shift: TShiftState;

X, Y: Integer);

begin

mouseX := X;

mouseY := Y;

secondDeleteMenu.Check(X, Y);

MenuBox.Invalidate;

end;

procedure TSecondDeleteFm.MenuBoxClick(Sender: TObject);

begin

secondDeleteMenu.ClickCheck(mouseX, mouseY);

end;

procedure TSecondDeleteFm.ShowHideTimerTimer(Sender: TObject);

begin

ShowHide(ShowHideTimer, self, DeleteFm);

end;

procedure TSecondDeleteFm.dbCbZoo3Click(Sender: TObject);

begin

if (dbCbZoo3.KeyValue <> 0) then

begin

qryShowPic.Close;

qryShowPic.ParamByName('id\_zoo').AsInteger := qryShowZoo3.FieldByName('id\_zoo').AsInteger;

qryShowPic.Open;

if (qryShowPic.RecordCount <> 0) then

begin

qryShowPic.First;

imgZoo.Hide;

imgZoo.Picture.LoadFromFile(dir + qryShowPic.FieldByName('Path').AsString);

secondDeleteMenu.menuItems[6].visible := true;

secondDeleteMenu.menuItems[7].visible := true;

currPic := 1;

end

else

begin

secondDeleteMenu.menuItems[6].visible := false;

secondDeleteMenu.menuItems[7].visible := false;

end;

end;

end;

procedure TSecondDeleteFm.dbCbZoo2Click(Sender: TObject);

begin

if (dbCbZoo2.KeyValue <> 0) then

begin

qryShowAnim2.Close;

qryShowAnim2.ParamByName('id\_Zoo').AsInteger := qryShowZoo4.FieldByName('id\_zoo').AsInteger;

qryShowAnim2.Open;

end;

end;

procedure TSecondDeleteFm.dbCbZoo4Click(Sender: TObject);

begin

if (dbCbZoo4.KeyValue <> 0) then

begin

qryShowAnim.Close;

qryShowAnim.ParamByName('id\_zoo').AsInteger := qryShowZoo3.FieldByName('id\_zoo').AsInteger;

qryShowAnim.Open;

end;

end;

procedure TSecondDeleteFm.dbCbAnim2Click(Sender: TObject);

begin

if (dbCbAnim2.KeyValue <> 0) then

begin

qryShowMat.Close;

qryShowMat.ParamByName('id\_link').AsInteger := qryShowAnim.FieldByName('id\_link').AsInteger;

qryShowMat.Open;

if (qryShowMat.RecordCount <> 0) then

begin

qryShowMat.First;

if (qryShowMat.FieldByName('type').AsInteger = 0) then

begin

imgAnim.Show;

videoPanel.Hide;

imgAnim.Picture.LoadFromFile(dir + qryShowMat.FieldByName('Path').AsString);

end

else

begin

imgAnim.Hide;

videoPanel.Show;

MP.FileName := dir + qryShowMat.FieldByName('Path').AsString;

MP.Open;

MP.DisplayRect := videoPanel.ClientRect;

MP.Play;

end;

currMat := 1;

secondDeleteMenu.menuItems[8].visible := true;

secondDeleteMenu.menuItems[9].visible := true;

end

else

begin

secondDeleteMenu.menuItems[8].visible := false;

secondDeleteMenu.menuItems[9].visible := false;

end;

end;

end;

end.

unit SecondUpdate;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, ExtCtrls, DB, DBTables, Grids, DBGrids, MenuUnit, StdCtrls,

DBCtrls, TimerUnit, ComCtrls;

type

TSecondUpdateFm = class(TForm)

MenuBox: TPaintBox;

ShowHideTimer: TTimer;

DBGrdZoo: TDBGrid;

qryShowZoo: TQuery;

ds1: TDataSource;

bvl1: TBevel;

edtUpdate: TEdit;

qryShow: TQuery;

ds2: TDataSource;

dbCbShow: TDBLookupComboBox;

lbl1: TLabel;

lbl2: TLabel;

lbl3: TLabel;

bvl2: TBevel;

DBGrdAnim: TDBGrid;

qryShowAnim: TQuery;

ds3: TDataSource;

DBGrdSpecies: TDBGrid;

lbl4: TLabel;

bvl3: TBevel;

qryShowSpecies: TQuery;

ds4: TDataSource;

dbmmoZooDescription: TDBMemo;

DBMmoNote: TDBMemo;

dbmmoDescription: TDBMemo;

dtp: TDateTimePicker;

mem: TMemo;

procedure MenuBoxPaint(Sender: TObject);

procedure MenuBoxMouseMove(Sender: TObject; Shift: TShiftState;

X, Y: Integer);

procedure MenuBoxClick(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure DBGrdZooDrawColumnCell(Sender: TObject; const Rect: TRect;

DataCol: Integer; Column: TColumn; State: TGridDrawState);

procedure LetKeyPress(Sender: TObject; var Key: Char);

procedure NumKeyPress(Sender: TObject; var Key: Char);

procedure DBGrdZooColExit(Sender: TObject);

procedure ShowHideTimerTimer(Sender: TObject);

procedure FormShow(Sender: TObject);

procedure FormHide(Sender: TObject);

procedure DBGrdAnimDrawColumnCell(Sender: TObject; const Rect: TRect;

DataCol: Integer; Column: TColumn; State: TGridDrawState);

procedure DBGrdAnimColExit(Sender: TObject);

procedure DBGrdSpeciesDrawColumnCell(Sender: TObject;

const Rect: TRect; DataCol: Integer; Column: TColumn;

State: TGridDrawState);

procedure DBGrdSpeciesColExit(Sender: TObject);

procedure DBGrdZooCellClick(Column: TColumn);

procedure DBGrdAnimCellClick(Column: TColumn);

procedure DBGrdSpeciesCellClick(Column: TColumn);

procedure edtUpdateKeyDown(Sender: TObject; var Key: Word;

Shift: TShiftState);

procedure dbmmoZooDescriptionClick(Sender: TObject);

procedure memExit(Sender: TObject);

procedure DBMmoNoteClick(Sender: TObject);

procedure dbmmoDescriptionClick(Sender: TObject);

procedure memKeyDown(Sender: TObject; var Key: Word;

Shift: TShiftState);

private

*{ Private declarations }*

public

*{ Public declarations }*

end;

var

SecondUpdateFm: TSecondUpdateFm;

implementation

uses Update, Message;

var mouseX: integer;

mouseY: integer;

secondUpdateMenu: TMyMenu;

menuItems: array of TMyMenuItem;

n: integer;

zooEdit: boolean;

animEdit: boolean;

speciesEdit: boolean;

{$R \*.dfm}

procedure BackToUpdateMenu();

begin

SecondUpdateFm.ShowHideTimer.Enabled := true;;

end;

procedure TSecondUpdateFm.MenuBoxPaint(Sender: TObject);

begin

secondUpdateMenu.DrawMenu(MenuBox.Canvas);

end;

procedure TSecondUpdateFm.MenuBoxMouseMove(Sender: TObject; Shift: TShiftState;

X, Y: Integer);

begin

mouseX := X;

mouseY := Y;

secondUpdateMenu.Check(X, Y);

MenuBox.Invalidate;

end;

procedure TSecondUpdateFm.MenuBoxClick(Sender: TObject);

begin

secondUpdateMenu.ClickCheck(mouseX, mouseY);

end;

procedure TSecondUpdateFm.FormCreate(Sender: TObject);

var bg: TBitmap;

begin

n := 1;

bg := TBitmap.Create();

bg.LoadFromFile(ExtractFilePath(Application.ExeName) + 'Pictures\BackGround.bmp');

secondUpdateMenu := TMyMenu.Create(n, bg);

SetLength(menuItems, n);

menuItems[0].x := 770;

menuItems[0].y := 520;

menuItems[0].size := 15;

menuItems[0].color := clSilver;

menuItems[0].name := 'Орто;

menuItems[0].visible := true;

menuItems[0].onClick := BackToUpdateMenu;

secondUpdateMenu.Add(menuItems[0]);

SecondUpdateFm.DoubleBuffered := true;

end;

procedure TSecondUpdateFm.DBGrdZooDrawColumnCell(Sender: TObject;

const Rect: TRect; DataCol: Integer; Column: TColumn;

State: TGridDrawState);

begin

if (gdFocused in State) then

begin

if (column.Field.FieldName = 'Title') or (column.Field.FieldName = 'City') then

begin

with edtUpdate do

begin

Left := Rect.Left + DbGrdZoo.Left;

Top := Rect.Top + DBGrdZoo.Top;

Width := Rect.Right - Rect.Left;

Height := Rect.Bottom - Rect.Top;

Clear;

Text := column.Field.Text;

OnKeyPress := LetKeyPress;

Visible := true;

end;

end

else if (column.Field.FieldName = 'Area') then

begin

with edtUpdate do

begin

Left := Rect.Left + DbGrdZoo.Left;

Top := Rect.Top + DBGrdZoo.Top;

Width := Rect.Right - Rect.Left;

Height := Rect.Bottom - Rect.Top;

Clear;

Text := column.Field.Text;

OnKeyPress := NumKeyPress;

Visible := true;

end;

end

else if (column.Field.FieldName = 'Country') then

begin

with dbCbShow do

begin

Left := Rect.Left + DbGrdZoo.Left;

Top := Rect.Top + DBGrdZoo.Top;

Width := Rect.Right - Rect.Left;

Height := Rect.Bottom - Rect.Top;

qryShow.Close;

KeyField := 'id\_country';

ListField := 'country';

KeyValue := qryShowZoo.FieldByName('id\_country').AsInteger;

with qryShow do

begin

Close;

SQL.Clear;

SQL.Add('select \* from database\countries');

Open;

end;

Visible := true;

end;

end;

end;

end;

procedure TSecondUpdateFm.LetKeyPress(Sender: TObject; var Key: Char);

begin

if (Key in ['0'..'9']) then

begin

Key := #0;

end;

end;

procedure TSecondUpdateFm.NumKeyPress(Sender: TObject; var Key: Char);

begin

if (Key in ['0'..'9'] + [#8]) = false then

begin

Key := #0;

end;

end;

procedure TSecondUpdateFm.DBGrdZooColExit(Sender: TObject);

var qry: TQuery;

begin

if ((DBGrdZoo.SelectedField.FieldName = 'Title') or

(DBGrdZoo.SelectedField.FieldName = 'City') or

(DBGrdZoo.SelectedField.FieldName = 'Area')) and

(zooEdit) then

begin

if (edtUpdate.Text <> DBGrdZoo.SelectedField.Text) then

begin

qry := TQuery.Create(self);

try

with qry do

begin

SQL.Clear;

SQL.Add('update database\Zoos');

SQL.Add('set ' + DBGrdZoo.SelectedField.FieldName + ' = "' + edtUpdate.Text + '"');

SQL.Add('where (id\_zoo = ' + IntToStr(qryShowZoo.FieldByName('id\_zoo').AsInteger) + ')');

ExecSQL;

qryShowZoo.Close;

qryShowZoo.Open;

qryShowAnim.Close;

qryShowAnim.Open;

qryShowSpecies.Close;

qryShowSpecies.Open;

end;

finally

qry.Free;

end;

end;

edtUpdate.Visible := false;

end

else if (DBGrdZoo.SelectedField.FieldName = 'Country') and (zooEdit) then

begin

qryShow.Open;

if (qryShow.FieldByName('id\_country').AsInteger <>

qryShowZoo.FieldByName('id\_country').AsInteger) then

begin

qry := TQuery.Create(self);

try

with qry do

begin

SQL.Clear;

SQL.Add('update database\Zoos');

SQL.Add('set id\_country = ' + IntToStr(qryShow.FieldByName('id\_country').AsInteger));

SQL.Add('where (id\_zoo = ' + IntToStr(qryShowZoo.FieldByName('id\_zoo').AsInteger) + ')');

ExecSQL;

qryShowZoo.Close;

qryShowZoo.Open;

qryShowAnim.Close;

qryShowAnim.Open;

qryShowSpecies.Close;

qryShowSpecies.Open;

end;

finally

qry.Free;

end;

end;

dbCbShow.Visible := false;

qryShow.Close;

end;

end;

procedure TSecondUpdateFm.ShowHideTimerTimer(Sender: TObject);

begin

ShowHide(ShowHideTimer, self, UpdateFm);

end;

procedure TSecondUpdateFm.FormShow(Sender: TObject);

begin

qryShowZoo.Open;

qryShowAnim.Open;

qryShowSpecies.Open;

zooEdit := false;

animEdit := false;

speciesEdit := false;

end;

procedure TSecondUpdateFm.FormHide(Sender: TObject);

begin

qryShowZoo.Close;

qryShowAnim.Close;

qryShowSpecies.Close;

end;

procedure TSecondUpdateFm.DBGrdAnimDrawColumnCell(Sender: TObject;

const Rect: TRect; DataCol: Integer; Column: TColumn;

State: TGridDrawState);

begin

if (gdFocused in State) then

begin

if (column.Field.FieldName = 'Name') then

begin

with edtUpdate do

begin

Left := Rect.Left + DbGrdAnim.Left;

Top := Rect.Top + DbGrdAnim.Top;

Width := Rect.Right - Rect.Left;

Height := Rect.Bottom - Rect.Top;

Clear;

Text := column.Field.Text;

OnKeyPress := LetKeyPress;

Visible := true;

end;

end

else if (column.Field.FieldName = 'Name\_1') then

begin

with dbCbShow do

begin

Left := Rect.Left + DbGrdAnim.Left;

Top := Rect.Top + DbGrdAnim.Top;

Width := Rect.Right - Rect.Left;

Height := Rect.Bottom - Rect.Top;

qryShow.Close;

KeyField := 'id\_species';

ListField := 'Name';

KeyValue := qryShowAnim.FieldByName('id\_species').AsInteger;

with qryShow do

begin

Close;

SQL.Clear;

SQL.Add('select \* from database\species');

Open;

end;

Visible := true;

end;

end

else if (column.Field.FieldName = 'Title') then

begin

with dbCbShow do

begin

Left := Rect.Left + DbGrdAnim.Left;

Top := Rect.Top + DbGrdAnim.Top;

Width := Rect.Right - Rect.Left;

Height := Rect.Bottom - Rect.Top;

qryShow.Close;

KeyField := 'id\_zoo';

ListField := 'Title';

KeyValue := qryShowZoo.FieldByName('id\_zoo').AsInteger;

with qryShow do

begin

Close;

SQL.Clear;

SQL.Add('select \* from database\zoos');

Open;

end;

Visible := true;

end;

end

else if (column.Field.FieldName = 'RegDate') then

begin

with dtp do

begin

Left := Rect.Left + DbGrdAnim.Left;

Top := Rect.Top + DbGrdAnim.Top;

Width := Rect.Right - Rect.Left;

Height := Rect.Bottom - Rect.Top;

Date := qryShowAnim.FieldByName('RegDate').AsDateTime;

Visible := true;

end;

end;

end;

end;

procedure TSecondUpdateFm.DBGrdAnimColExit(Sender: TObject);

var qry: TQuery;

begin

if (DbGrdAnim.SelectedField.FieldName = 'Name') and (animEdit) then

begin

if (edtUpdate.Text <> DBGrdAnim.SelectedField.Text) then

begin

qry := TQuery.Create(self);

try

with qry do

begin

SQL.Clear;

SQL.Add('update database\Link');

SQL.Add('set Name = "' + edtUpdate.Text + '"');

SQL.Add('where (id\_link = ' + IntToStr(qryShowAnim.FieldByName('id\_link').AsInteger) + ')');

ExecSQL;

qryShowZoo.Close;

qryShowZoo.Open;

qryShowAnim.Close;

qryShowAnim.Open;

qryShowSpecies.Close;

qryShowSpecies.Open;

end;

finally

qry.Free;

end;

end;

edtUpdate.Visible := false;

end

else if (DbGrdAnim.SelectedField.FieldName = 'Name\_1') and (animEdit) then

begin

qryShow.Open;

if (qryShow.FieldByName('Name').AsString <>

qryShowAnim.FieldByName('Name\_1').AsString) then

begin

qry := TQuery.Create(self);

try

with qry do

begin

SQL.Clear;

SQL.Add('update database\Link');

SQL.Add('set id\_species = ' + IntToStr(qryShow.FieldByName('id\_species').AsInteger));

SQL.Add('where (id\_link = ' + IntToStr(qryShowAnim.FieldByName('id\_link').AsInteger) + ')');

ExecSQL;

qryShowZoo.Close;

qryShowZoo.Open;

qryShowAnim.Close;

qryShowAnim.Open;

qryShowSpecies.Close;

qryShowSpecies.Open;

end;

finally

qry.Free;

end;

end;

dbCbShow.Visible := false;

qryShow.Close;

end

else if (DbGrdAnim.SelectedField.FieldName = 'Title') and (animEdit) then

begin

qryShow.Open;

if (qryShow.FieldByName('Title').AsString <>

qryShowAnim.FieldByName('Title').AsString) then

begin

qry := TQuery.Create(self);

try

with qry do

begin

SQL.Clear;

SQL.Add('update database\Link');

SQL.Add('set id\_zoo = ' + IntToStr(qryShow.FieldByName('id\_zoo').AsInteger));

SQL.Add('where (id\_link = ' + IntToStr(qryShowAnim.FieldByName('id\_link').AsInteger) + ')');

ExecSQL;

qryShowZoo.Close;

qryShowZoo.Open;

qryShowAnim.Close;

qryShowAnim.Open;

qryShowSpecies.Close;

qryShowSpecies.Open;

end;

finally

qry.Free;

end;

end;

dbCbShow.Visible := false;

qryShow.Close;

end

else if (DbGrdAnim.SelectedField.FieldName = 'RegDate') and (animEdit) then

begin

if (dtp.Date <>

qryShowAnim.FieldByName('RegDate').AsDateTime) then

begin

qry := TQuery.Create(self);

try

with qry do

begin

SQL.Clear;

SQL.Add('update database\Link');

SQL.Add('set RegDate = "' + DateToStr(dtp.Date) + '"');

SQL.Add('where (id\_link = ' + IntToStr(qryShowAnim.FieldByName('id\_link').AsInteger) + ')');

ExecSQL;

qryShowZoo.Close;

qryShowZoo.Open;

qryShowAnim.Close;

qryShowAnim.Open;

qryShowSpecies.Close;

qryShowSpecies.Open;

end;

finally

qry.Free;

end;

end;

dtp.visible := false;

end;

end;

procedure TSecondUpdateFm.DBGrdSpeciesDrawColumnCell(Sender: TObject;

const Rect: TRect; DataCol: Integer; Column: TColumn;

State: TGridDrawState);

begin

if (gdFocused in State) then

begin

if (column.Field.FieldName = 'Name') then

begin

with edtUpdate do

begin

Left := Rect.Left + DBGrdSpecies.Left;

Top := Rect.Top + DBGrdSpecies.Top;

Width := Rect.Right - Rect.Left;

Height := Rect.Bottom - Rect.Top;

Clear;

Text := column.Field.Text;

OnKeyPress := LetKeyPress;

Visible := true;

end;

end

else if (column.Field.FieldName = 'Habitat') then

begin

with dbCbShow do

begin

Left := Rect.Left + DBGrdSpecies.Left;

Top := Rect.Top + DBGrdSpecies.Top;

Width := Rect.Right - Rect.Left;

Height := Rect.Bottom - Rect.Top;

qryShow.Close;

KeyField := 'id\_habitat';

ListField := 'habitat';

KeyValue := qryShowSpecies.FieldByName('id\_habitat').AsInteger;

with qryShow do

begin

Close;

SQL.Clear;

SQL.Add('select \* from database\habitats');

Open;

end;

Visible := true;

end;

end

else if (column.Field.FieldName = 'FoodType') then

begin

with dbCbShow do

begin

Left := Rect.Left + DBGrdSpecies.Left;

Top := Rect.Top + DBGrdSpecies.Top;

Width := Rect.Right - Rect.Left;

Height := Rect.Bottom - Rect.Top;

qryShow.Close;

KeyField := 'id\_foodtype';

ListField := 'foodtype';

KeyValue := qryShowSpecies.FieldByName('id\_foodtype').AsInteger;

with qryShow do

begin

Close;

SQL.Clear;

SQL.Add('select \* from database\foodtypes');

Open;

end;

Visible := true;

end;

end

else if (column.Field.FieldName = 'Climate') then

begin

with dbCbShow do

begin

Left := Rect.Left + DBGrdSpecies.Left;

Top := Rect.Top + DBGrdSpecies.Top;

Width := Rect.Right - Rect.Left;

Height := Rect.Bottom - Rect.Top;

qryShow.Close;

KeyField := 'id\_climate';

ListField := 'climate';

KeyValue := qryShowSpecies.FieldByName('id\_climate').AsInteger;

with qryShow do

begin

Close;

SQL.Clear;

SQL.Add('select \* from database\climates');

Open;

end;

Visible := true;

end;

end;

end;

end;

procedure TSecondUpdateFm.DBGrdSpeciesColExit(Sender: TObject);

var qry: TQuery;

begin

if (DBGrdSpecies.SelectedField.FieldName = 'Name') and (speciesEdit) then

begin

if (edtUpdate.Text <> DBGrdSpecies.SelectedField.Text) then

begin

qry := TQuery.Create(self);

try

with qry do

begin

SQL.Clear;

SQL.Add('update database\species');

SQL.Add('set Name = "' + edtUpdate.Text + '"');

SQL.Add('where (id\_species = ' + IntToStr(qryShowSpecies.FieldByName('id\_species').AsInteger) + ')');

ExecSQL;

qryShowZoo.Close;

qryShowZoo.Open;

qryShowAnim.Close;

qryShowAnim.Open;

qryShowSpecies.Close;

qryShowSpecies.Open;

end;

finally

qry.Free;

end;

end;

edtUpdate.Visible := false;

end

else if (DBGrdSpecies.SelectedField.FieldName = 'Habitat') and (speciesEdit) then

begin

qryShow.Open;

if (qryShow.FieldByName('Habitat').AsString <>

qryShowSpecies.FieldByName('Habitat').AsString) then

begin

qry := TQuery.Create(self);

try

with qry do

begin

SQL.Clear;

SQL.Add('update database\Habitat');

SQL.Add('set id\_habitat = ' + IntToStr(qryShow.FieldByName('id\_habitat').AsInteger));

SQL.Add('where (id\_link = ' + IntToStr(qryShowSpecies.FieldByName('id\_habitat').AsInteger) + ')');

ExecSQL;

qryShowZoo.Close;

qryShowZoo.Open;

qryShowAnim.Close;

qryShowAnim.Open;

qryShowSpecies.Close;

qryShowSpecies.Open;

end;

finally

qry.Free;

end;

end;

dbCbShow.Visible := false;

qryShow.Close;

end

else if (DBGrdSpecies.SelectedField.FieldName = 'FoodType') and (speciesEdit) then

begin

qryShow.Open;

if (qryShow.FieldByName('FoodType').AsString <>

qryShowSpecies.FieldByName('FoodType').AsString) then

begin

qry := TQuery.Create(self);

try

with qry do

begin

SQL.Clear;

SQL.Add('update database\Species');

SQL.Add('set id\_foodType = ' + IntToStr(qryShow.FieldByName('id\_foodType').AsInteger));

SQL.Add('where (id\_species = ' + IntToStr(qryShowSpecies.FieldByName('id\_species').AsInteger) + ')');

ExecSQL;

qryShowZoo.Close;

qryShowZoo.Open;

qryShowAnim.Close;

qryShowAnim.Open;

qryShowSpecies.Close;

qryShowSpecies.Open;

end;

finally

qry.Free;

end;

end;

dbCbShow.Visible := false;

qryShow.Close;

end

else if (DBGrdSpecies.SelectedField.FieldName = 'Climate') and (speciesEdit) then

begin

qryShow.Open;

if (qryShow.FieldByName('Climate').AsString <>

qryShowSpecies.FieldByName('Climate').AsString) then

begin

qry := TQuery.Create(self);

try

with qry do

begin

SQL.Clear;

SQL.Add('update database\Species');

SQL.Add('set id\_climate = ' + IntToStr(qryShow.FieldByName('id\_climate').AsInteger));

SQL.Add('where (id\_species = ' + IntToStr(qryShowSpecies.FieldByName('id\_species').AsInteger) + ')');

ExecSQL;

qryShowZoo.Close;

qryShowZoo.Open;

qryShowAnim.Close;

qryShowAnim.Open;

qryShowSpecies.Close;

qryShowSpecies.Open;

end;

finally

qry.Free;

end;

end;

dbCbShow.Visible := false;

qryShow.Close;

end;

end;

procedure TSecondUpdateFm.DBGrdZooCellClick(Column: TColumn);

begin

zooEdit := true;

animEdit := false;

speciesEdit := false;

end;

procedure TSecondUpdateFm.DBGrdAnimCellClick(Column: TColumn);

begin

zooEdit := false;

animEdit := true;

speciesEdit := false;

end;

procedure TSecondUpdateFm.DBGrdSpeciesCellClick(Column: TColumn);

begin

zooEdit := false;

animEdit := false;

speciesEdit := true;

end;

procedure TSecondUpdateFm.edtUpdateKeyDown(Sender: TObject; var Key: Word;

Shift: TShiftState);

begin

if (key = VK\_RETURN) then

begin

if (zooEdit) then

DBGrdZooColExit(DBGrdZoo)

else if (animEdit) then

DBGrdAnimColExit(DBGrdAnim)

else if (speciesEdit) then

DBGrdSpeciesColExit(DBGrdSpecies);

end;

end;

procedure TSecondUpdateFm.dbmmoZooDescriptionClick(Sender: TObject);

begin

zooEdit := true;

if (zooEdit) and (not mem.Visible) then

begin

with mem do

begin

Left := dbmmoZooDescription.Left;

Top := dbmmoZooDescription.Top;

Width := dbmmoZooDescription.Width;

Height := dbmmoZooDescription.Height;

Text := dbmmoZooDescription.Text;

visible := true;

SetFocus;

end;

end;

animEdit := false;

speciesEdit := false;

end;

procedure TSecondUpdateFm.memExit(Sender: TObject);

var qry: TQuery;

begin

if (zooEdit) then

begin

if (mem.Text <> dbmmoZooDescription.Text) then

begin

if (Length(mem.Text) <> 0) then

begin

qry := TQuery.Create(SecondUpdateFm);

try

qry.Close;

qry.SQL.Add('update database\zoos');

qry.SQL.Add('set ZooDescription = "' + mem.Text + '"');

qry.SQL.Add('where id\_zoo = ' + IntToStr(qryShowZoo.FieldByName('id\_zoo').AsInteger));

qry.ExecSQL;

MessageFm.mess := 'Зпс ордкиоаа';

MessageFm.ShowModal;

qryShowZoo.Close;

qryShowZoo.Open;

finally

qry.Free;

end;

end

else

begin

MessageFm.mess := 'Пл н зплео';

MessageFm.ShowModal;

end;

end;

end

else if (animEdit) then

begin

if (mem.Text <> DBMmoNote.Text) then

begin

if (Length(mem.Text) <> 0) then

begin

qry := TQuery.Create(SecondUpdateFm);

try

qry.Close;

qry.SQL.Add('update database\link');

qry.SQL.Add('set Note = "' + mem.Text + '"');

qry.SQL.Add('where id\_link = ' + IntToStr(qryShowAnim.FieldByName('id\_link').AsInteger));

qry.ExecSQL;

MessageFm.mess := 'Зпс ордкиоаа';

MessageFm.ShowModal;

qryShowAnim.Close;

qryShowAnim.Open;

finally

qry.Free;

end;

end

else

begin

MessageFm.mess := 'Пл н зплео';

MessageFm.ShowModal;

end;

end;

end

else if (speciesEdit) then

begin

if (mem.Text <> dbmmoDescription.Text) then

begin

if (Length(mem.Text) <> 0) then

begin

qry := TQuery.Create(SecondUpdateFm);

try

qry.Close;

qry.SQL.Add('update database\species');

qry.SQL.Add('set Description = "' + mem.Text + '"');

qry.SQL.Add('where id\_species = ' + IntToStr(qryShowSpecies.FieldByName('id\_species').AsInteger));

qry.ExecSQL;

MessageFm.mess := 'Зпс ордкиоаа';

MessageFm.ShowModal;

qryShowSpecies.Close;

qryShowSpecies.Open;

finally

qry.Free;

end;

end

else

begin

MessageFm.mess := 'Пл н зплео';

MessageFm.ShowModal;

end;

end;

end;

zooEdit := false;

animEdit := false;

speciesEdit := false;

mem.Visible := false;

end;

procedure TSecondUpdateFm.DBMmoNoteClick(Sender: TObject);

begin

animEdit := true;

if (animEdit) and (not mem.Visible) then

begin

with mem do

begin

Left := DBMmoNote.Left;

Top := DBMmoNote.Top;

Width := DBMmoNote.Width;

Height := DBMmoNote.Height;

Text := DBMmoNote.Text;

visible := true;

SetFocus;

end;

end;

zooEdit := false;

speciesEdit := false;

end;

procedure TSecondUpdateFm.dbmmoDescriptionClick(Sender: TObject);

begin

speciesEdit := true;

if (speciesEdit) and (not mem.Visible) then

begin

with mem do

begin

Left := dbmmoDescription.Left;

Top := dbmmoDescription.Top;

Width := dbmmoDescription.Width;

Height := dbmmoDescription.Height;

Text := dbmmoDescription.Text;

visible := true;

SetFocus;

end;

end;

zooEdit := false;

animEdit := false;

end;

procedure TSecondUpdateFm.memKeyDown(Sender: TObject; var Key: Word;

Shift: TShiftState);

begin

if (Key = VK\_RETURN) then

begin

mem.OnExit(self);

end;

end;

end.

unit ShowMaterials;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, ExtCtrls, MPlayer, DB, DBTables, MenuUnit;

type

TShowMaterialsFm = class(TForm)

MenuBox: TPaintBox;

bvl6: TBevel;

imgZoo: TImage;

qryShowPicture: TQuery;

qryShowMaterial: TQuery;

videoPanel: TPanel;

MP: TMediaPlayer;

bvl5: TBevel;

imgAnimZoo: TImage;

procedure FormCreate(Sender: TObject);

procedure MenuBoxPaint(Sender: TObject);

procedure MenuBoxMouseMove(Sender: TObject; Shift: TShiftState;

X, Y: Integer);

procedure MenuBoxClick(Sender: TObject);

procedure FormShow(Sender: TObject);

private

*{ Private declarations }*

public

*{ Public declarations }*

end;

var

ShowMaterialsFm: TShowMaterialsFm;

implementation

var showMenu: TMyMenu;

menuItems: array of TMyMenuItem;

n: integer;

mouseX: integer;

mouseY: integer;

currPic: integer;

currMat: integer;

dir: string;

{$R \*.dfm}

procedure PriorZooPic();

begin

with ShowMaterialsFm do

begin

if (currPic <> 0) then

begin

qryShowPicture.Prior;

Dec(currPic);

imgZoo.Picture.LoadFromFile(dir +

qryShowPicture.FieldByName('Path').AsString);

end;

end;

end;

procedure NextZooPic();

begin

with ShowMaterialsFm do

begin

if (currPic <> qryShowPicture.RecordCount) then

begin

qryShowPicture.Next;

inc(currPic);

imgZoo.Picture.LoadFromFile(dir + qryShowPicture.FieldByName('Path').AsString);

end;

end;

end;

procedure PriorMaterial();

begin

with ShowMaterialsFm do

begin

if (currMat <> 0) then

begin

qryShowMaterial.Prior;

Dec(currMat);

if (qryShowMaterial.FieldByName('type').AsInteger = 0) then

begin

imgAnimZoo.Show;

videoPanel.Hide;

imgAnimZoo.Picture.LoadFromFile(dir + qryShowMaterial.FieldByName('Path').AsString);

MP.Close;

MP.FileName := '';

MP.Hide;

end

else

begin

imgAnimZoo.Hide;

videoPanel.Show;

MP.Show;

imgAnimZoo.Picture.Bitmap.FreeImage;

MP.FileName := dir + qryShowMaterial.FieldByName('Path').AsString;

MP.Open;

MP.Play;

MP.DisplayRect := videoPanel.ClientRect;

end;

end;

end;

end;

procedure NextMaterial();

begin

with ShowMaterialsFm do

begin

if (currMat <> qryShowMaterial.RecordCount) then

begin

qryShowMaterial.Next;

inc(currMat);

if (qryShowMaterial.FieldByName('type').AsInteger = 0) then

begin

imgAnimZoo.Show;

videoPanel.Hide;

imgAnimZoo.Picture.LoadFromFile(dir + qryShowMaterial.FieldByName('Path').AsString);

MP.Close;

MP.FileName := '';

end

else

begin

imgAnimZoo.Hide;

videoPanel.Show;

imgAnimZoo.Picture.Bitmap.FreeImage;

MP.FileName := dir + qryShowMaterial.FieldByName('Path').AsString;

MP.Open;

MP.Play;

MP.DisplayRect := videoPanel.ClientRect;

end;

end;

end;

end;

procedure TShowMaterialsFm.FormCreate(Sender: TObject);

var bg: TBitmap;

i: integer;

begin

n := 4;

SetLength(menuItems, n);

bg := TBitmap.Create;

bg.LoadFromFile(ExtractFilePath(Application.ExeName) + 'Pictures\BackGround.bmp');

showMenu := TMyMenu.Create(n, bg);

menuItems[0].x := 100;

menuItems[0].y := 263;

menuItems[0].size := 15;

menuItems[0].color := clSilver;

menuItems[0].name := '<';

menuItems[0].visible := true;

menuItems[0].onClick := PriorZooPic;

menuItems[1].x := 200;

menuItems[1].y := 263;

menuItems[1].size := 15;

menuItems[1].color := clSilver;

menuItems[1].name := '>';

menuItems[1].visible := true;

menuItems[1].onClick := NextZooPic;

menuItems[2].x := 100;

menuItems[2].y := 520;

menuItems[2].size := 15;

menuItems[2].color := clSilver;

menuItems[2].name := '<';

menuItems[2].visible := true;

menuItems[2].onClick := PriorMaterial;

menuItems[3].x := 200;

menuItems[3].y := 520;

menuItems[3].size := 15;

menuItems[3].color := clSilver;

menuItems[3].name := '>';

menuItems[3].visible := true;

menuItems[3].onClick := NextMaterial;

for i := 0 to n - 1 do

begin

showMenu.Add(menuItems[i]);

end;

dir := ExtractFilePath(Application.ExeName);

ShowMaterialsFm.DoubleBuffered := true;

MenuBox.Invalidate;

end;

procedure TShowMaterialsFm.MenuBoxPaint(Sender: TObject);

begin

showMenu.DrawMenu(MenuBox.Canvas);

end;

procedure TShowMaterialsFm.MenuBoxMouseMove(Sender: TObject; Shift: TShiftState;

X, Y: Integer);

begin

mouseX := X;

mouseY := Y;

showMenu.Check(X, Y);

MenuBox.Invalidate;

end;

procedure TShowMaterialsFm.MenuBoxClick(Sender: TObject);

begin

showMenu.ClickCheck(mouseX, mouseY);

end;

procedure TShowMaterialsFm.FormShow(Sender: TObject);

begin

currPic := 1;

currMat := 1;

end;

end.

unit ShowZoos;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, MenuUnit, TimerUnit, DB, DBTables, Grids, DBGrids, DBCtrls,

ExtCtrls, StdCtrls, MPlayer, OleServer, WordXP;

type

TShowZoosFm = class(TForm)

MenuBox: TPaintBox;

ShowHideTimer: TTimer;

dbCbCountry: TDBLookupComboBox;

dbGrdZoo: TDBGrid;

qryShowCountry: TQuery;

qryShowZoo: TQuery;

qryShowAnim: TQuery;

ds1: TDataSource;

ds2: TDataSource;

ds3: TDataSource;

dbGrdAnim: TDBGrid;

dbGrdAnim2: TDBGrid;

bvl1: TBevel;

bvl2: TBevel;

bvl3: TBevel;

bvl4: TBevel;

lbl1: TLabel;

lbl2: TLabel;

lbl3: TLabel;

dbmmoZoo: TDBMemo;

dbmmoAnim: TDBMemo;

dbmmoSpecies: TDBMemo;

ExtensionTimer: TTimer;

NarrowingTimer: TTimer;

ShowTimer: TTimer;

wordApp: TWordApplication;

wordDoc: TWordDocument;

procedure FormCreate(Sender: TObject);

procedure MenuBoxPaint(Sender: TObject);

procedure MenuBoxMouseMove(Sender: TObject; Shift: TShiftState; X,

Y: Integer);

procedure MenuBoxClick(Sender: TObject);

procedure dbCbCountryClick(Sender: TObject);

procedure FormShow(Sender: TObject);

procedure dbGrdZooCellClick(Column: TColumn);

procedure ShowHideTimerTimer(Sender: TObject);

procedure ExtensionTimerTimer(Sender: TObject);

procedure NarrowingTimerTimer(Sender: TObject);

procedure FormHide(Sender: TObject);

procedure dbGrdAnimCellClick(Column: TColumn);

procedure ShowTimerTimer(Sender: TObject);

private

*{ Private declarations }*

public

*{ Public declarations }*

end;

var

ShowZoosFm: TShowZoosFm;

implementation

uses

MainMenu, ShowMaterials, Search, ComObj, Message, DateUtils;

var

n: integer;

mouseX: integer;

mouseY: Integer;

showMenu: TMyMenu;

menuItems:array of TMyMenuItem;

dir: string;

minused: integer;

{$R \*.dfm}

procedure BackToMainMenu();

begin

ShowZoosFm.NarrowingTimer.Enabled := true;

ShowZoosFm.ShowHideTimer.Enabled := true;

end;

procedure FormExtension();

begin

ShowZoosFm.ExtensionTimer.Enabled := true;

showMenu.menuItems[1].visible := false;

end;

procedure FormNarrowing();

begin

ShowZoosFm.NarrowingTimer.Enabled := true;

showMenu.menuItems[2].visible := false;

end;

procedure ToSearchMenu();

begin

ShowZoosFm.ShowTimer.Enabled := true;

end;

procedure ExportToWord();

var myParagraph: Paragraph;

indexDoc: OleVariant;

indexParagraph: OleVariant;

i: integer;

picId: integer;

LinkToFile: OleVariant;

SaveWithDocument: OleVariant;

FileName: OleVariant;

begin

with ShowZoosFm do

begin

try

indexDoc := 1;

indexParagraph := 0;

wordApp.Connect;

wordApp.Documents.Add(EmptyParam, EmptyParam, EmptyParam, EmptyParam);

wordDoc.ConnectTo(wordApp.ActiveDocument);

wordDoc.Paragraphs.Add(EmptyParam);

indexParagraph := indexParagraph + 1;

myParagraph := wordDoc.Paragraphs.Item(indexParagraph);

myParagraph.Range.Text := qryShowZoo.FieldByName('Title').AsString;

myParagraph.Alignment := wdAlignParagraphCenter;

myParagraph.Range.Font.Size := 14;

myParagraph.Range.Font.Bold := 1;

myParagraph.Range.Font.Name := 'Times new roman';

wordDoc.Paragraphs.Add(EmptyParam);

indexParagraph := indexParagraph + 1;

myParagraph := wordDoc.Paragraphs.Item(indexParagraph);

myParagraph.Range.Text := 'Ияжвтоо ' +

qryShowAnim.FieldByName('Name').AsString;

myParagraph.Range.Font.Bold := 0;

myParagraph.Range.Font.Size := 12;

myParagraph.Range.Font.Name := 'Times new roman';

myParagraph.Alignment := wdAlignParagraphLeft;

wordDoc.Paragraphs.Add(EmptyParam);

indexParagraph := indexParagraph + 1;

myParagraph := wordDoc.Paragraphs.Item(indexParagraph);

myParagraph.Range.Text := 'Дт ргсрци ' +

DateToStr(qryShowAnim.FieldByName('RegDate').AsDateTime);

wordDoc.Paragraphs.Add(EmptyParam);

indexParagraph := indexParagraph + 1;

myParagraph := wordDoc.Paragraphs.Item(indexParagraph);

myParagraph.Range.Text := 'Вд ' +

qryShowAnim.FieldByName('Name\_1').AsString;

wordDoc.Paragraphs.Add(EmptyParam);

myParagraph.Range.Text := 'Сеаоиаи: ' +

qryShowAnim.FieldByName('Habitat').AsString;

indexParagraph := indexParagraph + 1;

myParagraph := wordDoc.Paragraphs.Item(indexParagraph);

wordDoc.Paragraphs.Add(EmptyParam);

indexParagraph := indexParagraph + 1;

myParagraph := wordDoc.Paragraphs.Item(indexParagraph);

myParagraph.Range.Text := 'Тпптня ' +

qryShowAnim.FieldByName('FoodType').AsString;

myParagraph.Alignment := wdAlignParagraphLeft;

wordDoc.Paragraphs.Add(EmptyParam);

indexParagraph := indexParagraph + 1;

myParagraph := wordDoc.Paragraphs.Item(indexParagraph);

myParagraph.Range.Text := 'Киа мсапоиаи: ' +

qryShowAnim.FieldByName('Climate').AsString;

myParagraph.Alignment := wdAlignParagraphLeft;

wordDoc.Paragraphs.Add(EmptyParam);

indexParagraph := indexParagraph + 1;

myParagraph := wordDoc.Paragraphs.Item(indexParagraph);

myParagraph.Range.Text := 'Оиаи жвтоо ';

myParagraph.Alignment := wdAlignParagraphCenter;

wordDoc.Paragraphs.Add(EmptyParam);

indexParagraph := indexParagraph + 1;

myParagraph := wordDoc.Paragraphs.Item(indexParagraph);

myParagraph.Range.Text := dbmmoAnim.Text;

myParagraph.Alignment := wdAlignParagraphLeft;

wordDoc.Paragraphs.Add(EmptyParam);

indexParagraph := indexParagraph + 1;

myParagraph := wordDoc.Paragraphs.Item(indexParagraph);

myParagraph.Range.Text := 'Оиаи вд: ';

myParagraph.Alignment := wdAlignParagraphCenter;

wordDoc.Paragraphs.Add(EmptyParam);

indexParagraph := indexParagraph + 1;

myParagraph := wordDoc.Paragraphs.Item(indexParagraph);

myParagraph.Range.Text := dbmmoSpecies.Text;

myParagraph.Alignment := wdAlignParagraphLeft;

with ShowMaterialsFm.qryShowMaterial do

begin

if (RecordCount <> 0) then

begin

wordDoc.Paragraphs.Add(EmptyParam);

indexParagraph := indexParagraph + 1;

myParagraph := wordDoc.Paragraphs.Item(indexParagraph);

myParagraph.Range.Text := 'Фтгаи: ';

myParagraph.Alignment := wdAlignParagraphCenter;

picId := FieldByName('id\_material').AsInteger;

LinkToFile := false;

SaveWithDocument := true;

First;

for i := 1 to RecordCount do

begin

if (FieldByName('type').AsInteger = 0) then

begin

FileName := ExtractFilePath(Application.ExeName) +

FieldByName('Path').AsString;

wordDoc.Paragraphs.Add(EmptyParam);

indexParagraph := indexParagraph + 1;

myParagraph := wordDoc.Paragraphs.Item(indexParagraph);

myParagraph.Range.InlineShapes.

AddPicture(FileName, LinkToFile, SaveWithDocument, EmptyParam);

end;

Next;

end;

First;

for i := 1 to RecordCount do

begin

if (picId = FieldByName('id\_material').AsInteger) then

break

else

Next;

end;

end;

end;

wordApp.Visible := true;

finally

wordApp.Disconnect;

end;

end;

end;

procedure TShowZoosFm.FormCreate(Sender: TObject);

var bg: TBitmap;

i: integer;

begin

minused := 0;

dir := ExtractFilePath(Application.ExeName);

n := 5;

bg := TBitmap.Create;

bg.LoadFromFile(ExtractFilePath(Application.ExeName) + 'Pictures\BackGround.bmp');

showMenu := TMyMenu.Create(n, bg);

SetLength(menuItems, n);

menuItems[0].x := 600;

menuItems[0].y := 520;

menuItems[0].size := 15;

menuItems[0].color := clSilver;

menuItems[0].name := 'Орто;

menuItems[0].visible := true;

menuItems[0].onClick := BackToMainMenu;

menuItems[1].x := 600;

menuItems[1].y := 235;

menuItems[1].size := 15;

menuItems[1].color := clSilver;

menuItems[1].name := '>>>';

menuItems[1].visible := true;

menuItems[1].onClick := FormExtension;

menuItems[2].x := 600;

menuItems[2].y := 235;

menuItems[2].size := 15;

menuItems[2].color := clSilver;

menuItems[2].name := '<<<';

menuItems[2].visible := false;

menuItems[2].onClick := FormNarrowing;

menuItems[3].x := 56;

menuItems[3].y := 520;

menuItems[3].size := 15;

menuItems[3].color := clSilver;

menuItems[3].name := 'Пик;

menuItems[3].visible := true;

menuItems[3].onClick := ToSearchMenu;

menuItems[4].x := 300;

menuItems[4].y := 520;

menuItems[4].size := 15;

menuItems[4].color := clSilver;

menuItems[4].name := 'ЭсотвWord';

menuItems[4].visible := true;

menuItems[4].onClick := ExportToWord;

for i := 0 to n - 1 do

begin

showMenu.Add(menuItems[i]);

end;

ShowZoosFm.DoubleBuffered := true;

MenuBox.Invalidate;

end;

procedure TShowZoosFm.MenuBoxPaint(Sender: TObject);

begin

showMenu.DrawMenu(MenuBox.Canvas);

end;

procedure TShowZoosFm.MenuBoxMouseMove(Sender: TObject; Shift: TShiftState;

X, Y: Integer);

begin

mouseX := X;

mouseY := Y;

showMenu.Check(X, Y);

MenuBox.Invalidate;

end;

procedure TShowZoosFm.MenuBoxClick(Sender: TObject);

begin

showMenu.ClickCheck(mouseX, mouseY);

end;

procedure TShowZoosFm.dbCbCountryClick(Sender: TObject);

begin

if (dbCbCountry.KeyValue <> 0) then

begin

qryShowZoo.Close;

qryShowZoo.ParamByName('id\_country').AsInteger :=

qryShowCountry.FieldByName('id\_country').AsInteger;

qryShowZoo.Open;

dbGrdZooCellClick(dbGrdZoo.Columns.Items[0]);

dbGrdAnimCellClick(dbGrdAnim.Columns.Items[0]);

end

else

dbCbCountry.KeyValue := 0;

end;

procedure TShowZoosFm.FormShow(Sender: TObject);

begin

minused := 0;

ShowZoosFm.Left := 141 + 175;

showMenu.menuItems[1].visible := true;

showMenu.menuItems[2].visible := false;

qryShowCountry.Close;

qryShowCountry.SQL.Clear;

qryShowCountry.SQL.Add('select \* from database\countries');

qryShowCountry.Open;

qryShowZoo.Close;

qryShowZoo.SQL.Clear;

qryShowZoo.SQL.Add('select \* from database\zoos z');

qryShowZoo.SQL.Add('where (id\_Country = :id\_country)');

qryShowZoo.ParamByName('id\_country').AsInteger := 0;

qryShowZoo.Open;

qryShowAnim.Close;

qryShowAnim.SQL.Clear;

qryShowAnim.SQL.Add('select \* from database\link l');

qryShowAnim.SQL.Add('inner join database\species s');

qryShowAnim.SQL.Add('on l.id\_species = s.id\_species');

qryShowAnim.SQL.Add('inner join database\habitats h');

qryShowAnim.SQL.Add('on s.id\_habitat = h.id\_habitat');

qryShowAnim.SQL.Add('inner join database\FoodTypes f');

qryShowAnim.SQL.Add('on s.id\_foodtype = f.id\_foodtype');

qryShowAnim.SQL.Add('inner join database\Climates c');

qryShowAnim.SQL.Add('on s.id\_climate = c.id\_climate');

qryShowAnim.SQL.Add('where (l.id\_zoo = :id\_zoo)');

qryShowAnim.ParamByName('id\_zoo').AsInteger := 0;

qryShowAnim.Open;

qryShowCountry.First;

dbCbCountry.KeyValue := qryShowCountry.FieldByName('id\_country').AsInteger;

dbcbCountryClick(self);

dbGrdZooCellClick(dbGrdAnim.Columns.Items[0]);

dbGrdAnimCellClick(dbGrdAnim.Columns.Items[0]);

end;

procedure TShowZoosFm.dbGrdZooCellClick(Column: TColumn);

begin

with ShowMaterialsFm do

begin

qryShowPicture.Close;

qryShowPicture.ParamByName('id\_zoo').AsInteger :=

qryShowZoo.FieldByName('id\_zoo').AsInteger;

qryShowPicture.Open;

qryShowPicture.First;

if (qryShowPicture.RecordCount <> 0) then

imgZoo.Picture.LoadFromFile(dir +

qryShowPicture.FieldByName('Path').AsString)

else

imgZoo.Picture.LoadFromFile(dir + '\pictures\NoPhoto.jpg');

end;

qryShowAnim.Close;

qryShowAnim.ParamByName('id\_zoo').AsInteger :=

qryShowZoo.FieldByName('id\_zoo').AsInteger;

qryShowAnim.Open;

end;

procedure TShowZoosFm.ShowHideTimerTimer(Sender: TObject);

begin

ShowHide(ShowHideTimer, ShowZoosFm, MainMenuFm);

end;

procedure TShowZoosFm.ExtensionTimerTimer(Sender: TObject);

begin

if (minused <> 175) then

begin

if (minused = 0) then

begin

BlockInput(true);

end;

minused := minused + 5;

ShowZoosFm.Left := ShowZoosFm.Left - 5;

end

else

begin

if (ShowMaterialsFm.AlphaBlendValue <> 255) then

begin

if (ShowMaterialsFm.AlphaBlendValue = 0) then

begin

ShowMaterialsFm.Show;

ShowMaterialsFm.Left := ShowMaterialsFm.Left + 370;

end;

ShowMaterialsFm.AlphaBlendValue := ShowMaterialsFm.AlphaBlendValue + 5;

end

else

begin

BlockInput(false);

showMenu.menuItems[2].visible := true;

ExtensionTimer.Enabled := false;

ShowZoosFm.MenuBox.Enabled := true;

ShowMaterialsFm.MenuBox.Enabled := true;

end;

end;

end;

procedure TShowZoosFm.NarrowingTimerTimer(Sender: TObject);

begin

if (ShowMaterialsFm.AlphaBlendValue <> 0) then

begin

if (ShowMaterialsFM.AlphaBlendValue = 255) then

BlockInput(true);

ShowMaterialsFm.AlphaBlendValue := ShowMaterialsFm.AlphaBlendValue - 5;

if (ShowMaterialsFm.AlphaBlendValue = 0) then

begin

ShowMaterialsFm.Left := ShowMaterialsFm.Left - 370;

ShowMaterialsFm.Hide;

end;

end

else

begin

if (minused <> 0) then

begin

minused := minused - 5;

ShowZoosFm.Left := ShowZoosFm.Left + 5;

end

else

begin

BlockInput(false);

NarrowingTimer.Enabled := false;

showMenu.menuItems[1].visible := true;

ShowZoosFm.MenuBox.Enabled := true;

ShowMaterialsFm.MenuBox.Enabled := true;

end;

end;

end;

procedure TShowZoosFm.FormHide(Sender: TObject);

begin

ShowMaterialsFm.MP.FileName := '';

ShowMaterialsFm.MP.Close;

qryShowCountry.Close;

qryShowZoo.Close;

qryShowAnim.Close;

ShowMaterialsFm.imgZoo.Picture.Bitmap.FreeImage;

ShowMaterialsFm.imgAnimZoo.Picture.Bitmap.FreeImage;

ShowMaterialsFm.MP.Close;

ShowMaterialsFm.MP.FileName := '';

end;

procedure TShowZoosFm.dbGrdAnimCellClick(Column: TColumn);

begin

with ShowMaterialsFm do

begin

imgAnimZoo.Picture.Bitmap.FreeImage;

MP.Close;

MP.FileName := '';

qryShowMaterial.Close;

qryShowMaterial.ParamByName('id\_link').AsInteger := qryShowAnim.FieldByName('id\_link').AsInteger;

qryShowMaterial.Open;

qryShowMaterial.First;

if (qryShowMaterial.RecordCount <> 0) then

if (qryShowMaterial.FieldByName('type').AsInteger = 0) then

begin

videoPanel.Hide;

imgAnimZoo.Show;

imgAnimZoo.Picture.LoadFromFile(dir +

qryShowMaterial.FieldByName('Path').AsString);

MP.Hide;

end

else

begin

MP.Show;

imgAnimZoo.Hide;

videoPanel.Show;

MP.FileName := dir + qryShowMaterial.FieldByName('Path').AsString;

MP.Open;

MP.DisplayRect := videoPanel.ClientRect;

end

else

imgAnimZoo.Picture.LoadFromFile(dir + 'Pictures\NoPhoto.jpg');

end;

end;

procedure TShowZoosFm.ShowTimerTimer(Sender: TObject);

begin

if (SearchFm.AlphaBlendValue <> 255) then

begin

if (SearchFm.AlphaBlendValue = 0) then

begin

TimerUnit.BlockInput(true);

SearchFm.Show;

end;

SearchFm.AlphaBlendValue := SearchFm.AlphaBlendValue + 5;

end

else

begin

ShowTimer.Enabled := false;

TimerUnit.BlockInput(false);

end;

end;

end.

unit TimerUnit;

interface

uses

Forms, ExtCtrls, Controls, Windows, Messages, jpeg, SysUtils;

procedure ShowHide(Sender: TTimer; Closable: TForm; Sliding: TForm);

var BlockInput : function(Block: BOOL): BOOL; stdcall;

implementation

uses MenuUnit;

var count: integer = 0;

function FuncAvail(dllName, funcName: string; var p: pointer): boolean;

var

lib: THandle;

begin

result := false;

p := nil;

if LoadLibrary(PChar(dllName)) = 0 then exit;

lib := GetModuleHandle(PChar(dllName)) ;

if lib <> 0 then

begin

p := GetProcAddress(lib, PChar(funcName)) ;

if p <> nil then Result := true;

end;

end;

procedure ShowHide(Sender: TTimer; Closable: TForm; Sliding: TForm);

begin

if (Closable.AlphaBlendValue <> 0) then

begin

if (Closable.AlphaBlendValue = 255) then

begin

FuncAvail('USER32.DLL', 'BlockInput', @BlockInput);

Sliding.Enabled := false;

Sliding.Show;

BlockInput(true);

end;

Closable.AlphaBlendValue := Closable.AlphaBlendValue - 5;

end

else

begin

if (Sliding.AlphaBlendValue <> 255) then

begin

Sliding.AlphaBlendValue := Sliding.AlphaBlendValue + 5;

end

else

begin

BlockInput(False);

Sliding.Enabled := true;

Closable.Hide;

Sender.Enabled := false;

end;

end;

end;

end.

unit Update;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, ExtCtrls, Grids, DBGrids, StdCtrls, DBCtrls, DB, DBTables, Menus,

MenuUnit, TimerUnit;

type

TUpdateFm = class(TForm)

MenuBox: TPaintBox;

bvl1: TBevel;

lbl1: TLabel;

RadGr1: TRadioGroup;

ShowHideTimer: TTimer;

procedure FormCreate(Sender: TObject);

procedure MenuBoxPaint(Sender: TObject);

procedure MenuBoxMouseMove(Sender: TObject; Shift: TShiftState;

X, Y: Integer);

procedure MenuBoxClick(Sender: TObject);

procedure ShowHideTimerTimer(Sender: TObject);

private

*{ Private declarations }*

public

*{ Public declarations }*

end;

var

UpdateFm: TUpdateFm;

implementation

uses

AdminMenu, FirstUpdate, Message, SecondUpdate;

var n: integer;

mouseX: integer;

mouseY: integer;

menuItems: array of TMyMenuItem;

updateMenu: TMyMenu;

form: TForm;

{$R \*.dfm}

procedure ToUpdate();

begin

form := FirstUpdateFm;

case (UpdateFm.RadGr1.ItemIndex) of

0:begin

FirstUpdate.idName := 'id\_country';

FirstUpdate.fieldName := 'country';

FirstUpdate.tableName := 'Countries';

FirstUpdateFm.LbShowUpdate.Caption := 'Вбрт срн: ';

form := FirstUpdateFm;

UpdateFm.ShowHideTimer.Enabled := true;

end;

1:begin

FirstUpdate.idName := 'id\_foodType';

FirstUpdate.fieldName := 'foodType';

FirstUpdate.tableName := 'FoodTypes';

FirstUpdateFm.LbShowUpdate.Caption := 'Вбрт тпптня ';

form := FirstUpdateFm;

UpdateFm.ShowHideTimer.Enabled := true;

end;

2:begin

FirstUpdate.idName := 'id\_climate';

FirstUpdate.fieldName := 'climate';

FirstUpdate.tableName := 'Climates';

FirstUpdateFm.LbShowUpdate.Caption := 'Вбрт киа: ';

form := FirstUpdateFm;

UpdateFm.ShowHideTimer.Enabled := true;

end;

3:begin

FirstUpdate.idName := 'id\_habitat';

FirstUpdate.fieldName := 'habitat';

FirstUpdate.tableName := 'Habitats';

FirstUpdateFm.LbShowUpdate.Caption := 'Вбрт сеуоиаи: ';

form := FirstUpdateFm;

UpdateFm.ShowHideTimer.Enabled := true;

end;

4:begin

form := SecondUpdateFm;

UpdateFm.ShowHideTimer.Enabled := true;

end;

else

MessageFm.mess := 'Н вба пнтрдкиоаи!';

messageFm.ShowModal;

end;

end;

procedure BackToAdmMenu();

begin

form := AdminMenuFm;

UpdateFm.ShowHideTimer.Enabled := true;

end;

procedure TUpdateFm.FormCreate(Sender: TObject);

var bg: TBitmap;

begin

n := 2;

bg := TBitmap.Create;

bg.LoadFromFile(ExtractFilePath(Application.ExeName) + 'Pictures\BackGround.bmp');

updateMenu := TMyMenu.Create(n, bg);

SetLength(menuItems, n);

menuItems[0].x := 160;

menuItems[0].y := 250;

menuItems[0].visible := true;

menuItems[0].color := clSilver;

menuItems[0].size := 15;

menuItems[0].name := 'Рдкиоаь;

menuItems[0].onClick := ToUpdate;

menuItems[1].x := 180;

menuItems[1].y := 280;

menuItems[1].visible := true;

menuItems[1].color := clSilver;

menuItems[1].size := 15;

menuItems[1].name := 'Орто;

menuItems[1].onClick := BackToAdmMenu;

updateMenu.Add(menuItems[0]);

updateMenu.Add(menuItems[1]);

UpdateFm.DoubleBuffered := true;

MenuBox.Invalidate;

end;

procedure TUpdateFm.MenuBoxPaint(Sender: TObject);

begin

updateMenu.DrawMenu(MenuBox.Canvas);

end;

procedure TUpdateFm.MenuBoxMouseMove(Sender: TObject; Shift: TShiftState;

X, Y: Integer);

begin

mouseX := X;

mouseY := Y;

updateMenu.Check(X, Y);

MenuBox.Invalidate;

end;

procedure TUpdateFm.MenuBoxClick(Sender: TObject);

begin

updateMenu.ClickCheck(mouseX, mouseY);

end;

procedure TUpdateFm.ShowHideTimerTimer(Sender: TObject);

begin

ShowHide(ShowHideTimer, UpdateFm, form);

end;

end.

program Zoos;

uses

Forms,

ScreenSaver in 'ScreenSaver.pas' *{ScreenSaverFm}*,

MainMenu in 'MainMenu.pas' *{MainMenuFm}*,

MenuUnit in 'MenuUnit.pas',

TimerUnit in 'TimerUnit.pas',

AdminMenu in 'AdminMenu.pas' *{AdminMenuFm}*,

Message in 'Message.pas' *{MessageFm}*,

Question in 'Question.pas' *{QuestionFm}*,

Add in 'Add.pas' *{AddFm}*,

ShowZoos in 'ShowZoos.pas' *{ShowZoosFm}*,

ShowMaterials in 'ShowMaterials.pas' *{ShowMaterialsFm}*,

Update in 'Update.pas' *{UpdateFm}*,

SecondDelete in 'SecondDelete.pas' *{SecondDeleteFm}*,

FirstUpdate in 'FirstUpdate.pas' *{FirstUpdateFm}*,

SecondUpdate in 'SecondUpdate.pas' *{SecondUpdateFm}*,

Search in 'Search.pas' *{SearchFm}*,

Delete in 'Delete.pas' *{DeleteFm}*,

FirstDelete in 'FirstDelete.pas' *{FirstDeleteFm}*,

Password in 'Password.pas' *{PasswordDlg}*;

{$R \*.res}

begin

Application.Initialize;

Application.Title := 'Зоак мр';

Application.CreateForm(TScreenSaverFm, ScreenSaverFm);

Application.CreateForm(TMainMenuFm, MainMenuFm);

Application.CreateForm(TAdminMenuFm, AdminMenuFm);

Application.CreateForm(TMessageFm, MessageFm);

Application.CreateForm(TQuestionFm, QuestionFm);

Application.CreateForm(TAddFm, AddFm);

Application.CreateForm(TShowZoosFm, ShowZoosFm);

Application.CreateForm(TShowMaterialsFm, ShowMaterialsFm);

Application.CreateForm(TUpdateFm, UpdateFm);

Application.CreateForm(TSecondDeleteFm, SecondDeleteFm);

Application.CreateForm(TFirstUpdateFm, FirstUpdateFm);

Application.CreateForm(TSecondUpdateFm, SecondUpdateFm);

Application.CreateForm(TSearchFm, SearchFm);

Application.CreateForm(TDeleteFm, DeleteFm);

Application.CreateForm(TFirstDeleteFm, FirstDeleteFm);

Application.CreateForm(TPasswordDlg, PasswordDlg);

Application.Run;

end.

**ПРИЛОЖЕНИЕ Б**

**(справочное)**

**Экранные формы**



Рисунок Б.1 – Форма заставки



Рисунок Б.2 – Форма меню пользователя

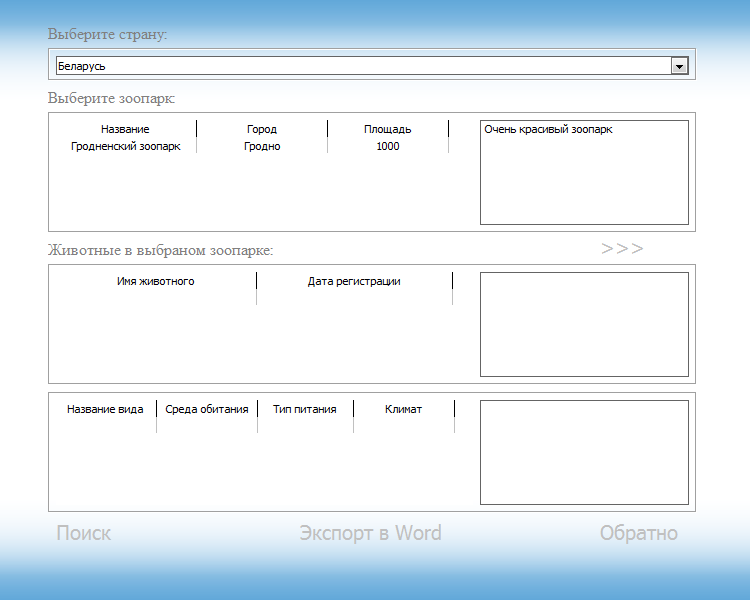


Рисунок Б.3 – Форма просмотра информации

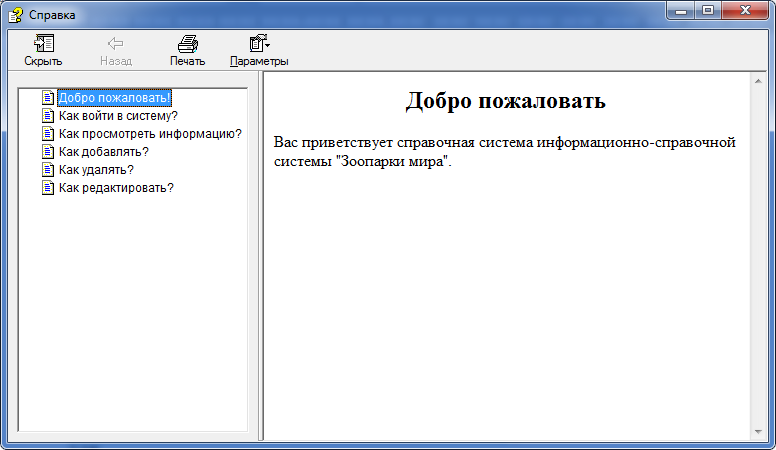


Рисунок Б.4 – Окно справочной системы

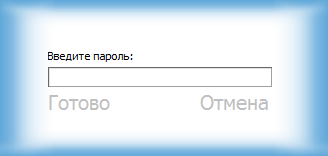


Рисунок Б.5 – Форма ввода пароля

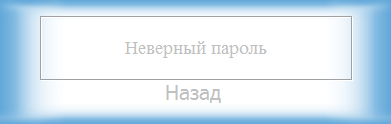


Рисунок Б.6 – Сообщение о неправильном пароле

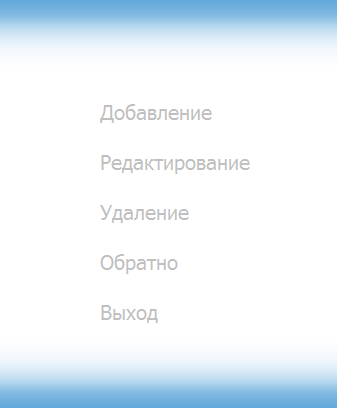


Рисунок Б.7 – Форма меню администратора

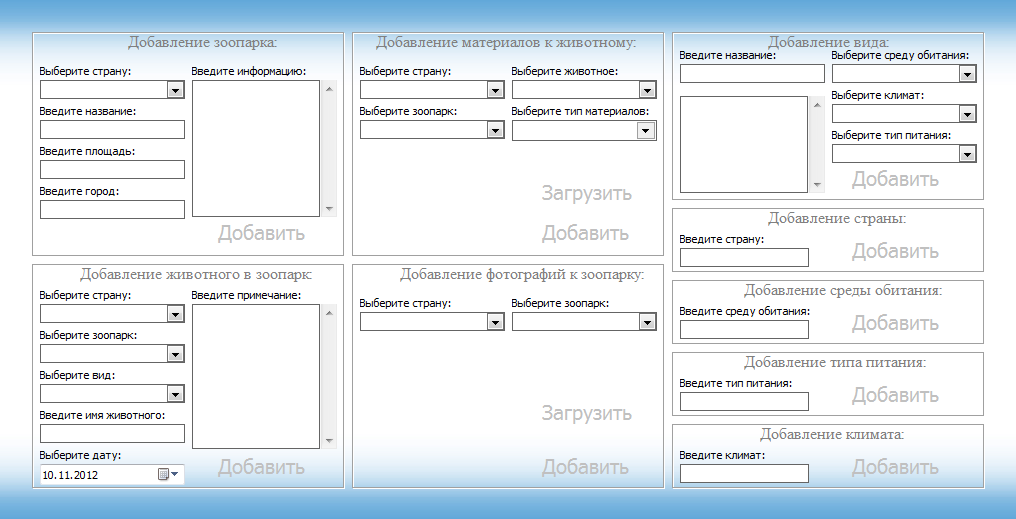


Рисунок Б.8 – Форма добавления

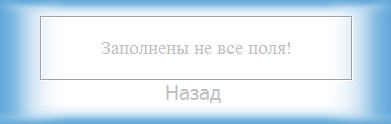


Рисунок Б.9 – Сообщение о незаполненных полях

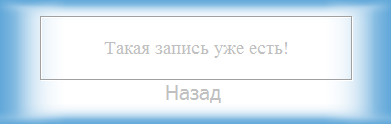


Рисунок Б.10 – Сообщение о существовании такой записи

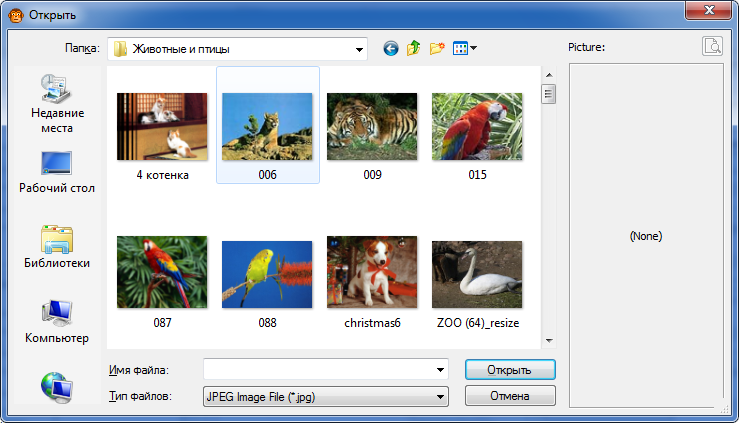


Рисунок Б.11 – Окно выбора фотографии и видеоматериала

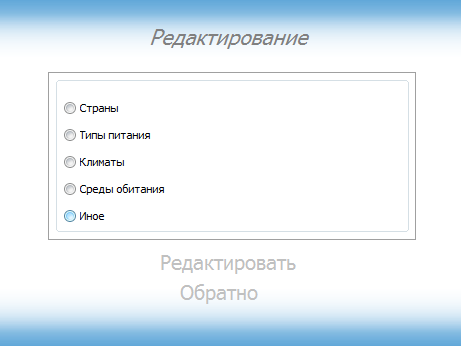


Рисунок Б.12 – Форма редактирования

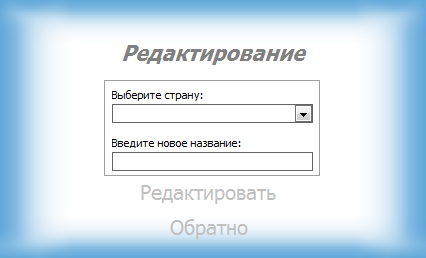


Рисунок Б.13 – Форма редактирования страны

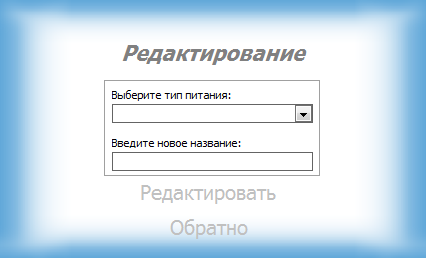


Рисунок Б.14 – Форма редактирования типа питания

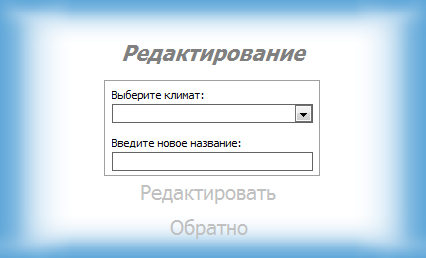


Рисунок Б.15 – Форма редактирования климата

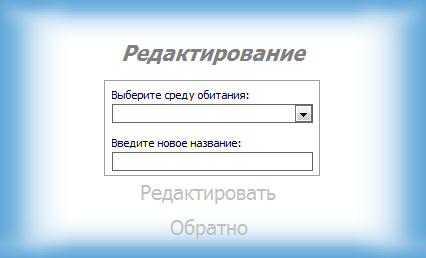


Рисунок Б.16 – Форма редактирования среды обитания

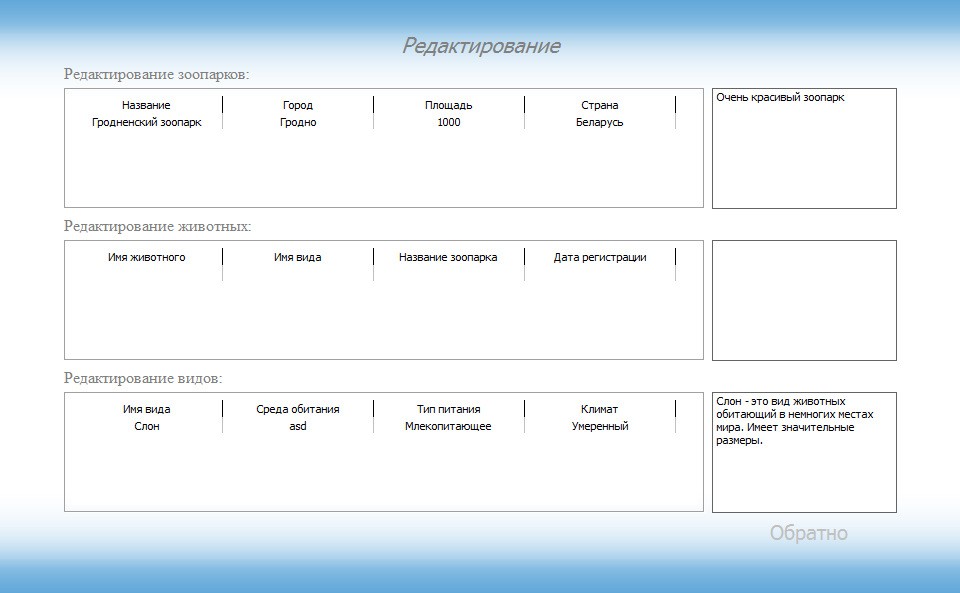


Рисунок Б.17 – Форма редактирования иных данных

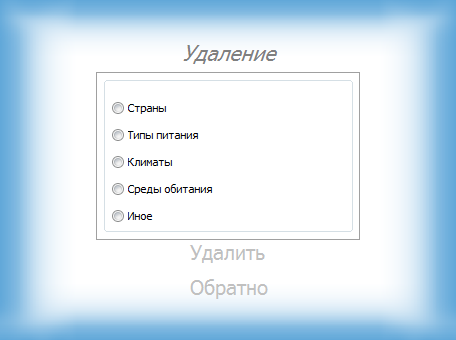


Рисунок Б.18 – Форма удаления

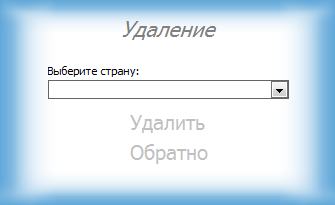


Рисунок Б.19 – Форма удаления страны

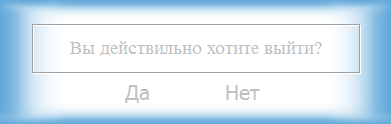


Рисунок Б.20 – Сообщение о подтверждении удаления

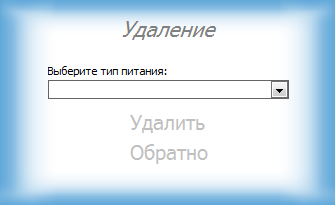


Рисунок Б.21 – Форма удаления типа питания

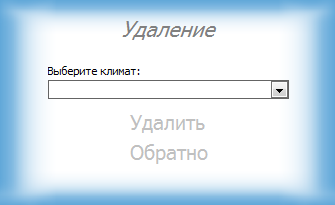


Рисунок Б.22 – Форма удаления климата

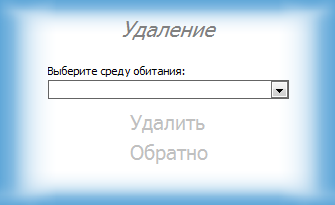


Рисунок Б.23 – Форма удаления среды обитания

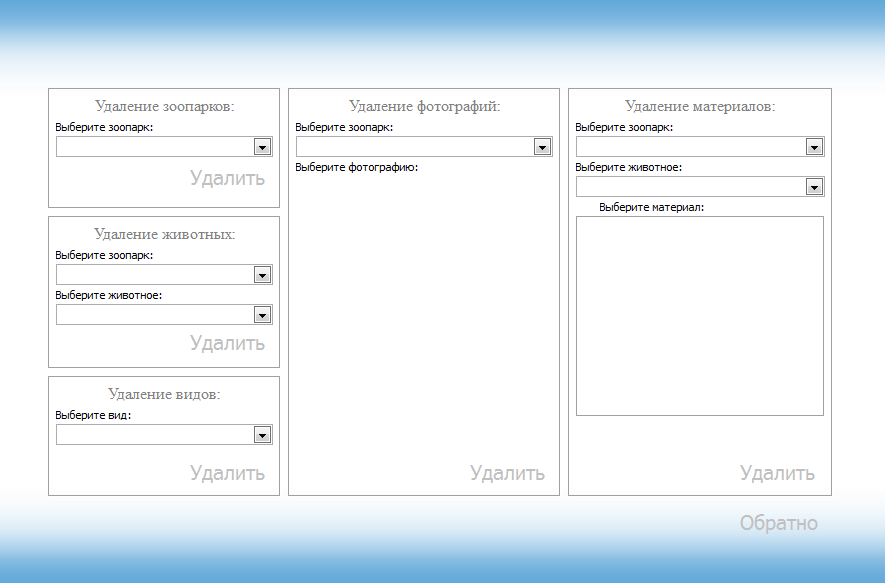


Рисунок Б.24 – Форма удаления иных данных

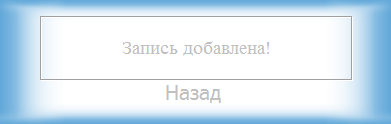


Рисунок Б.25 – Сообщение об успешном добавлении

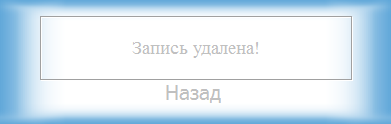


Рисунок Б.26 – Сообщение об успешном удалении